Optimal Quality of Service Control in Communication Systems

The research in this dissertation has been carried out in the context of the IOP GenCom project Service Optimization and Quality (SeQual), which is supported by the Dutch Ministry of Economic Affairs, Agriculture and Innovation via its agency Agentschap NL.

Printed by *Ipskamp Drukkers*, The Netherlands.

ISBN: 978-94-6259-029-8

VRIJE UNIVERSITEIT

Optimal Quality of Service Control in Communication Systems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Exacte Wetenschappen
op woensdag 12 februari 2014 om 11.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Joost Willem Bosman

geboren te Amsterdam

promotoren: prof.dr. R.D. van der Mei
prof.dr. R. Núñez-Queija

Dankwoord (Acknowledgments)

Het moment is daar, voor u ligt het werk van vier voorbijgevlogen jaren.

Dit werk was niet mogelijk geweest zonder mijn promotoren Rob van der Mei en Sindo Núñez Queija. Rob, jouw tomeloze emergie en optimisme heeft me geïnspireerd en veel geleerd. Je hebt me binnengehaald als stagair en me enthousiast gemaakt voor dit promotieonderzoek. Sindo, mij blijven de vele waardevolle middagen bij die we hebben besteed aan het kraken van moeilijke problemen. Ook wil ik de leden van leescommissie: Mark Squillante, Hans van den Berg, Sandjai Bhulai, Michel Mandjes en Erik Meeuwissen voor de grondige wijze waarop zij mijn proefschrift gereviewd hebben.

Ik wil Gerard Hoekstra en Sandjai Bhulai bedanken voor de vruchtbare samenwerking, die loopt vanaf het moment dat ik als stagair bij het CWI begon tot aan onze laatste publicatie die in deze dissertatie is verwerkt. Veel collega's hebben mij van waardevolle, wetenschappelijke input voorzien. In het bijzonder wil ik daarvoor Demeter Kiss, Yoni Nazarathy en Florian Simatos bedanken. Bert Zwart wil ik bedanken voor zijn inspiratie op zowel wetenschappenlijk als sportief gebied.

Ook ben ik dank verschuldigd aan Hans van den Berg, Erik Meeuwissen en Miroslav Živković voor een prettige samenwerking met TNO, waarvan de resultaten in deze dissertatie zijn verwerkt.

Het bezoek aan congressen heeft mij wereldwijd naar een aantal schitterende locaties gebracht. Tijdens mijn eerste congres in Toledo, vlak bij Madrid, werd de sfeer versterkt door de WK wedstrijden van het Nederlands elftal. Verder heb ik tijdelijk de winter kunnen vermijden door na een congres in Miami door Florida rond te trekken met als grote finale Key-West. Eén van de congressen heeft mij zelfs in Japan gebracht, waar de reis begon op een congres in Fukuoka en eindigde in een rondreis met de Shinkansen (hoge snelheidstreinen).

Goede herinneringen heb ik aan mijn kamergenoten Chrétien Verhoef en Arnoud den Boer. Ik dank Chrétien voor de avonturen die we beleefd hebben. Arnoud, bedankt voor je wetenschappelijke en filosofische gesprekken. Na de verbouwing van het CWI verhuisde ik naar de kamer van Jan-Pieter Dorsman, die een zeer bedreven wetenschapper is. Dat was ook goed te merken als de oplossing van een vraagstuk hem niet zinde ;). Een bron van goede wetenschap is een omgeving waar je je gemakkelijk voelt. Onze groep bestaat uit veel jonge wetenschappers die het goed met elkaar kunnen vinden. De gezamenlijke activiteiten bleven niet beperkt binnen de activiteitenruimte van het CWI gebouw. Zo maakten we mooie hardlooprondes langs het Flevopark en door Diemen. Bovendien gingen we regelmatig samen wat

drinken. Wat dat betreft heb ik goede herinneringen aan collega's: Martijn Onderwater, Martin van Buuren (onze vrolijke noot), Sihan Ding, Pieter van den Berg en de vele andere collega's in de Stochastics groep.

Op donderdagen is een groot deel van onze afdeling te vinden bij de OBP groep op de Vrije Universiteit. Ik wil Ger Koole bedanken voor het bieden van gastvrijheid op de VU en de mooie tochten die we hebben gemaakt door de Alpen.

In the winter of 2012 a research visit brought me to the department of Mark Squillante at the IBM Thomas J. Watson Research Center. The visit started very adventurous as my flight was amongst the first flights to New York after hurricane Sandy. Immediately when I started my research visit, I was involved in a challenging project together with Mayank Sharma, Yingdong Lu. Mark, thank you for the inspiring time that I spent at IBM.

Tenslotte wil ik mijn ouders bedanken. Jullie ambitie, zorg en geduld heeft mij geïnspireerd om mij te ontwikkelen tot wie ik nu ben.

Joost Bosman
Amsterdam, 2014

Table of Contents

1 Introduction

In a globally connected world, on-line services essentially operate in a 24/7 economy. The emergence of high-speed Internet, mobile communications and smart devices like smart phones and tablets provide people access to all kinds of services, anytime, anywhere. Moreover, both private and public organizations tend to migrate their administrative services to on-line environments. The Dutch government for example introduced an on-line identity service for governmental services to file tax returns, social security applications and enrollments for education. As a consequence, our modern society has become largely dependent on the availability of these services, while at any time the slightest disruptions in service are noticed and may have a huge impact on user experience and reputation.

At the same time, the structure of ICT systems has become more complex. In the last few years, we are witnessing a paradigm shift from the traditional information-oriented Internet into an Internet of Services (IoS), catalyzed by concepts like Service Oriented Architectures (SOA), Software as a Service, Platform as a Service, Infrastructure as a Service and Cloud Computing. This has opened up virtually unbounded possibilities for the creation of new and innovative services that facilitate business processes and improve the quality of life. A fundamental characteristic of the IoS is that new services may combine and integrate functionalities of other services. This often leads to complex and large chains of services offered by a multitude of third parties, each with its own business incentives.

Due to growing complexity and increasing dependence on services, reliability has become an issue of great importance. Providing reliable (i.e., available, trustworthy) and robust (i.e., resistant against system failures, cyber attacks, high-load and overload situations, flash crowds) ICT services has become crucial for our economy at large. These developments have raised the need for means to control the quality of complex large-scale ICT chains.

1.1 Goals

In current practice, quality for composite services is usually controlled on an ad-hoc basis, while the consequences of failures in service chains are often not well understood. A main concern is that, although such an approach might work for small chains, it will become unfeasible for future complex global-scale service chains. This raises the need for mechanisms that enable efficient usage of available shared

resources while preserving the desired Quality of Service (QoS) as perceived by the end user.

There are many optimization mechanisms available that could accomplish this. Examples of such mechanisms are response-time driven dynamic service composition, load balancing across parallel wireless networks, and play-out buffering in streaming media. The problem is that in general these mechanisms are not suitably tailored for the current and evolving information and communication systems. The controls and thresholds are often based on simple improvised rules. As a consequence, the enormous potential of QoS mechanisms to enhance service quality remains largely unexploited.

The main challenge faced in this thesis is how to effectively use QoS mechanisms for large-scale complex ICT systems with shared resources. To this end, we develop, analyze, optimize and evaluate quantitative models that capture the dynamics of QoS-control mechanisms and their implications on the user-perceived QoS. In doing so, our analyses ultimately lead to the development of scalable and robust algorithms, decision tables, and rules-of-thumb for the optimal use of QoS-control mechanisms.

1.2 Challenges

The development of efficient QoS mechanisms is complicated by the omnipresence of the phenomenon of uncertainty. Stochastic models are instrumental to capture such uncertainties and provide a basis for educated control of systems with uncertainty. One may distinguish the following three types of uncertainty.

Uncertainty about demand for resources. Most demand is driven by user behavior. We characterize three different time scales. On the timescale of years, developments like the emergence of cloud based services, and developments in multimedia drive an overall growth of demand over time. A key factor here is that bandwidth available for users is growing due to the evolution of new technologies e.g., Digital Subscriber Line (DSL), and third and fourth generation mobile networks. These factors contribute to a global growth, both in frequency and size of demand for resources. In medium long time scale (a few years or smaller), seasonality effects kick in, for example yearly, monthly, weekly, daily or even hourly patterns. Across small time scales (minutes, seconds or smaller), fluctuations are more unpredictable as effects like session duration behavior becomes visible.

It is important to note that all of the above fits within the notion of predictable user behavior. However, there are also many factors that are inherently unpredictable but may have a huge impact on resource availability (cyber attacks, flash crowds).

For this purpose, mechanisms are required that can respond to this unpredictable behavior and provide robustness to threats and undesired behavior.

Variability in resource availability (shared resources). Various factors contribute to variability in resource availability such as resource sharing, network or system failure, chaotic behavior, and temporary overload. For a majority of Internet resources, capacity is shared among the different users. As a result, in the perspective of the users, the availability of resource capacity varies. The level at which this is disturbing is determined by the elasticity of the service requirement. Elasticity is the level at which the data flow of a service can be slowed down or accelerated without impacting the perceived end result. For example, a video stream is barely elastic as eventually, when the available bandwidth decreases, the video stream will be distorted or play-out will stall. File transfers on the other hand, are elastic. Elasticity determines whether there is a need for minimal resource availability during the service. When demand exceeds the capacity for a short period of time, temporary overload may occur. In this sense, temporary overload is closely related to variability in demand for resources.

Not only shared demand but also the occurrence of chaotic behavior, network failure or system failure contribute to variability in resource availability. Chaotic behavior may for example be caused by unexpected interactions between systems, often due to misconfiguration. In worst cases misconfiguration causes network or system failures. To exemplify this, one could consider large cloud services offered by parties like Google or Amazon. In these services there is a need to dispatch demand according to geographic features of the requests. These demand volumes are so high that individual systems cannot handle all demand. In these cases, any configuration error has enormous consequences.

Limited information. Many existing models assume that the stochastic behavior of demand and resources is known. In practice, however this is rarely the case. Typically external parties at best have limited information about the internal behavior of a system. An issue of importance here is to what extent information is available to control models regarding the processes running in the system. Also external factors impact the challenge of limited information from system behavior. Systems possibly operate in changing environments driven by uncertain, unpredictable factors. To respond in a fashionable way, mechanisms are required that can adapt to these changes. The key challenge is partial observability. In partially observable systems, the stochastic nature and corresponding behavior of the (underlying) processes cannot be fully observed. For example one may only observe aggregated information about the system ($x + y$ instead of x and y), or information may only becomes available at course grained time intervals. In this setting, partial information approaches prove to be useful. Partial information approaches tend to recover as

much knowledge as possible about the unobservable stochastic nature of a system by using the information that can derived from past observations. In other words, not only the pure observations are used but also the order and age (in time units) of the observations may be used.

1.3 Overview of the dissertation

In **Chapter 2** we consider streaming media applications in an environment with shared resources. The shared nature of these resources causes fluctuations in the available bandwidth. Specifically, we study a tandem model consisting of two fluid queues. The first fluid queue models congestion due to resource sharing. Fluctuations are modeled by a Continuous Time Markov Chain (CTMC). The second buffer represents the play-out buffer.

We determine, by using extreme value theory, a proper choice for the initial play-out-buffer level, providing a given probabilistic guarantee on undisturbed playback. Our analysis is based on a result for the distribution of the maximum buffer level during a busy period. In this chapter we focus our analysis on the case a two-state CTMC. For this model we derive explicit expressions in terms of its parameters.

In **Chapter 3** we extend the results of Chapter 2 to a CTMC with arbitrary number of states. The complication here is that we need asymptotic properties of the inverse over a partition from a matrix exponential that represents the maximum level in a busy period. For the two state case this can be explicitly determined. In general however, this will lead to an intractable expression.

We are able to analyze the general case CTMC model. Using a spectral theory analysis approach, we examine the asymptotic properties and derive a characterization of the parameters of the asymptotic extreme value distribution. Using the results of Chapter 2 this leads to a proper choice for the initial play-out-buffer level for case with a multiple state CTMC.

In **Chapter 4** we consider routing of traffic from stations with concurrent (parallel) access to multiple wireless networks. Multi-path communication solutions provide a promising means to improve network performance in areas covered by multiple wireless access networks. To this end, we model the wireless networks by processor sharing nodes. By using the assumption of Poisson arrivals the system, we model the numbers of flows through the networks by a CTMC. Our goal is to minimize the expected transfer time of elastic data traffic by smartly dispatching the jobs to the networks, based on partial information about the numbers of foreground and background jobs in each of the nodes. Such a smart dispatching strategy is called a (deci-

sion) policy. In the case of full state information, the optimal dispatching policy can be derived via standard MDP-techniques, but for models with partial information an optimal solution is hard to obtain. An important requirement is that the routing algorithm is efficient, yet simple, easy-to-implement, scalable in the number of parallel networks and robust against changes in the parameter settings.

We propose a simple index rule for splitting traffic streams based on partial information, and benchmark the results against the optimal MDP solution in the case of full state information. We demonstrate by extensive simulations with real networks that our method performs extremely well under practical circumstances for a wide range of realistic parameter settings.

In **Chapter 5** we consider a general class of dynamic resource allocation problems within a stochastic optimal control framework. This class of problems arises in a wide variety of applications, each of which intrinsically involves resources of different types and demand with uncertainty and/or variability. Our goal is to dynamically allocate capacity for each resource type in order to serve the uncertain/variable demand and maximize the expected net-benefit based on the rewards and costs associated with the different resources. X. Gao, Y. Lu, M. Sharma and M. Squillante derived the optimal control policy within a singular control setting, which includes easily implementable algorithms for governing the dynamic adjustments to resource allocation capacities over time. The control setting uses a financial mathematics approach that hedges against future risks associated with resource allocation decisions and uncertain demand.

We have benchmarked this policy against other methods in the literature in a realistic setting. Accordingly, we developed a simulation environment in which experiments are constructed to demonstrate that this control policy is working extremely well. To make the setting more realistic we analyze Internet-traffic traces and fit these to a demand model that is used in the simulation experiments. Numerical experiments investigate various issues of both theoretical and practical interest, quantifying the significant benefits of our approach over alternative optimization approaches.

In **Chapter 6** we consider dynamic compositions of Web-services offered by third parties. We represent the composite Web-service as a (sequential) workflow of tasks. For each task within this workflow, a number of third-party service alternatives may be available. We assume that the third-party service (task) alternatives offer the same functionality at different price-quality levels. The execution of the workflow is controlled by an orchestrator that is programmed with a decision strategy. Service composition strategies can be either static, (i.e., according to a fixed set of decisions) or dynamic, (i.e., based on state information of the workflow). Our goal

is to find a dynamic strategy that maximizes the expected benefit for the composite service providers subject to an end-to-end response time objective.

We propose a dynamic programming approach for the optimization of the dynamic service selection strategy. To this end, we use the concept of response time budget. The response time budget is the time left, during the workflow execution, until the response time objective is violated. We conduct an extensive numerical study on a wide range of parameter values. The results demonstrate a significant gain in expected benefits while using our dynamic approach, just by simply using an easy implementable, pre-calculated lookup-table.

In **Chapter 7** we relax the assumptions of Chapter 6 and suppose that the response time distributions are unknown and have to be obtained empirically from response time observations during the execution of requests. Our objective is to obtain a dynamic control mechanism that is robust against changes in the environment it is operating in. Therefore we consider the situation where the response time distributions are changing over time. Moreover we consider the case where services break down at random and generate no response at all.

We propose a runtime control mechanism that dynamically optimizes service composition in real time by learning and adapting to changes in third party service response time behaviors. Accordingly, we adopted statistical tests to our setting. For demonstration of the usefulness of our approach we have implemented our control mechanism in a simulation. Moreover, we evaluate the influence of the control parameter settings on the effectiveness of the control mechanism.

A Fluid Model Analysis of Streaming Media in the Presence of Time-Varying Bandwidth

2

Over the past few years, the tremendous popularity of smart mobile end devices and services (like YouTube) has boosted the demand for streaming media applications offered via the Internet. One of the key requirements for the success of providers of such services is the ability to deliver services at competitive price-quality ratios. However, the Internet provides no more than best-effort service quality. Therefore, the packet streams generated by streaming media applications are distorted by fluctuations in the available bandwidth on the Internet, which may be significant over the duration of a typical streaming application (whose duration may range from a few minutes to tens of minutes). To cope with these distortions, play-out buffers temporarily store packets so as to reproduce the signal with a fixed delay offset (see Figure 2.1).

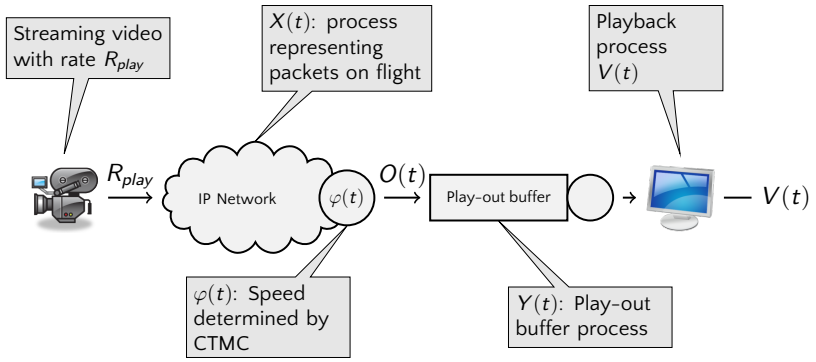


Figure 2.1: *Streaming video through an IP-network.*

Figure 2.1 clarifies the connection of the model described in Figure 2.2 with the application to streaming over an unreliable medium. For smooth reproduction of the packet stream the play-out buffer should not empty, as the stream will stall whenever packets do not arrive in time. For that reason, it is beneficial to start the play-out of a streaming media application *only when the play-out buffer content exceeds some safety threshold value*. In this context, our main goal is to determine a proper choice for the initial play-out-buffer level, providing a given probabilistic guarantee on undisturbed playback. Our objective in this chapter is to contribute to the understanding of the performance implications of the play-out-buffer settings for streaming applications over unreliable networks such as the Internet, by relating

[2] This chapter is based on [22].

the proper buffer level to network variability parameters. Congestion is modeled by a fluid queue with fixed input rate and output rate determined by a stochastic process that is modeled as a Continuous Time Markov Chain (CTMC). This CTMC represents the IP network dynamics that causes congestion and fluctuations in available bandwidth. If this model is applied to a real network, the CTMC parameters must be estimated in order to capture the network behavior. Our approach relies on a queuing-theoretical fluid model analysis.

The precise object of study in this chapter is a fluid model for constant bit-rate streaming media applications in the presence of bandwidth that varies over time (see Remark 2.6.1 for use of the model for variable bit rate applications). Motivated from Figure 2.1, we consider a tandem model consisting of two fluid queues. The first queue is a Markov Modulated fluid queue that models the congestion in the network caused by bandwidth fluctuations. The second buffer represents the play-out buffer.

2.1 Background

Buffer dimensioning for streaming video over variable rate networks has already received considerable attention in literature over the past two decades. Most work focused on balancing play-out buffer overflow and underrun probabilities, and develop dimensioning rules for the play-out buffer at the receiver end using analytic models. A particular large collection of work emerged in the 1990s. For example Bléfari-Melazzi et al. [15], and Kontovassilis et al. [71] determine the probability of overflow at the play-out buffer. This metric is particularly relevant for interactive video with stringent delay requirements, but less so for non-interactive streaming. More recently, play-out buffer engineering regained interest in the context of Voice over IP (VoIP), with popular examples such as Skype and Google Talk in Wu et al. [108]. Again, VoIP play-out buffer dimensioning must balance between conversational interactivity and speech quality. Proper dimensioning of the play-out buffer is known to have a decisive impact on conversation quality [108]. The *real time interactive* character of VoIP, however, poses again stringent restrictions on the buffer size, making the trade-off very different from non-interactive (video) streaming, which is the objective of this chapter. A third such example is in the context of closed-loop control for wireless streaming: Dua and Ambos [41], for example, investigate dynamic rules for play-out buffer management to avoid both the overflow of the play-out buffer and stalling of the streaming application. The setting studied in Kim et al. [70] is closest in nature to that in this chapter. Their approach, however, builds on a ``square root'' formula to approximate the throughput of TCP and the stalling probability is obtained through a fixed-point solution. Somewhat tangent to the above mentioned literature, there are works that concentrate on dy-

namic deterministic optimization, e.g. Tabrizi et al. [84], and Zhang et al. [115]. In our model, network unreliability is captured by a stochastic (Markovian) process and buffer dimensioning is tailored to the variability of the network.

Despite the large volume of literature devoted to play-out buffer dimensioning, the problem is still highly timely because of tremendous popularity of video streaming services such as YouTube. This popularity is catalyzed by two main developments. One is the continuing rise of streaming media usage on mobile devices, who suffer from highly unpredictable channel conditions, making an accurate buffer dimensioning rule crucial for viability of such services. Cisco's global mobile data traffic forecasts predict that mobile video will make up for 66% of all mobile data traffic in 2017, amounting to an approximate monthly 7 Exabytes of mobile video worldwide, from less than 1 Exabyte in 2013 [37]. Second, the market for video traffic over the Internet shows tremendous growth as well, in terms of numbers of users as well as in traffic volume. Cisco [36] predicts that in 2017, every second, nearly a million minutes of video content will cross the global IP network, making up for 69% of all consumer Internet traffic (from 57% in 2012). This number further increases to nearly 90% if video exchanged through peer-to-peer file sharing is included. Particularly relevant is *Internet video to TV*, which doubled in 2012 and continues to grow at a rapid pace, increasing fivefold by 2017.

Both in the context of wireless streaming and video on demand, a natural performance metric is the probability of uninterrupted video play out. The non-interactive nature of these services and the fact that memory is not the limiting factor on modern devices (naturally, mobile devices have much less memory, but videos played on mobile devices are streamed at a much lower bit rate also), make the memory usage a secondary consideration. The foremost important tradeoff is then between the initial play-out delay and the probability of stalling. We therefore set off to determine the *smallest* initial play-out delay (i.e. initial size of the play-out buffer) that gives a probabilistic guarantee on uninterrupted play out.

The above mentioned papers all focus on an engineering perspective. From a theoretical angle there is a considerable volume of related research too. Our modeling approach was already depicted in Figure 2.2: We will use a tandem of fluid queues (one with variable rate) to capture the most essential ingredients that determine the stalling probability (a detailed model description follows later). Fluid queues have proven to be a powerful modeling paradigm in a wide range of applications and have received much attention in literature. On one hand fluid model often capture the key characteristics that determine the performance of e.g. communication networks with complex packet-level dynamics (hiding largely irrelevant details), while on the other hand they remain mathematically tractable. Many analytic results have been obtained, and we refer to Scheinhardt [98], and Kulkarni [74] for excellent overviews of results on fluid queues that are directly relevant to our analysis. Asmussen and Bladt [6] propose a sample-path approach to study mean busy periods in Markov

Modulated fluid queues, and derive a simple way of calculating mean busy periods in terms of steady-state quantities. In [4], Asmussen shows that the probability of buffer overflow within a busy cycle has an exponential tail, gives an explicit expression for the Laplace Transform of the busy period and, moreover, derives several inequalities and approximations for the transient behaviour. Boxma and Dumas [23] study the busy period of a fluid queue fed by N ON/OFF sources with exponential OFF periods and heavy tailed activity durations (more specifically, with regularly varying activity duration distributions). Scheinhardt and Zwart [99] study a two-node tandem with gradual input, and compute the steady-state joint buffer-content distribution using martingale methods. Kulkarni and Tzenova [75] study a fluid queuing systems with different fluid-arrival rates governed by a CTMC and constant service rate. For this model, they derive a system of first-order non-homogeneous linear differential equations for the mean passage time. Sericola and Remiche [101] propose a method to analyse the maximum level and the hitting probabilities in a Markov driven fluid queue for various initial condition scenarios, allowing for both finite and infinite buffers. Their analysis leads to matrix differential Ricatti equations for which there is a unique solution. Asmussen [4] investigates a more general setting than the one considered in this chapter, which focuses on the streaming video setting. In our work we use an alternative matrix-theoretic analysis technique and obtain more explicit dimensioning rules than can be derived directly from specializing [4] to our model.

We derive a dimensioning rule for the play-out buffer, based on an extreme value distribution approximation. Our analysis is strongly motivated by the classical papers of Berman [13] and Iglehart [65]. Berman [13] studies the limiting distribution of the maximum in sequences of random variables satisfying certain dependence conditions. Iglehart [65] derived asymptotic distributions for the extreme value of the buffer content and the number of customers in the GI/G/1 queue. We refer to Asmussen [5] for an excellent survey on extreme-value theory for queues. This chapter provides an alternative approach for the analysis in Asmussen [4] specified to our model. Through our approach, we obtain more explicit results for the targeted dimensioning rules.

Our analysis proceeds as follows: We use results from [101] for the analysis of the maximum in a busy period. Furthermore, we show that the busy period maximum has an exponential tail and the maximum grows logarithmically. We apply a result on mean busy periods from [75] to obtain the mean expected cycle time. Next we apply an approach similar to [65] in order to show that the maximum buffer level converges to a Gumbel extreme value distribution. From this result the correct initial play-out buffer level can be estimated. As mentioned previously, our work shows strong similarities with [4]. Like us, Asmussen shows that the maximum fluid level grows logarithmically over time and under proper scaling converges to random variable with a Gumbel extreme value distribution. In this chapter we independently establish this result in a more intuitive manner. Based on this result, we derive an

explicit expression for the initial level of the play-out-buffer at which the play-out can best be started so as to guarantee undisturbed play-out with sufficient certainty.

2.2 Model

In our model we mimic a video stream that has fixed data rate R_{play}. Video is streamed through an IP network with fluctuating speed. From the IP network packets arrive to the play-out buffer with a rate that can take values from a finite set $\{s_i,\ i = 1, 2, \dots, n\}$. The actual output rate of the network is determined by a stochastic process $\varphi(t)$ that is modeled by an n-state CTMC. The CTMC has generator matrix T and state-space $\mathcal{S} = \{1, \dots, n\}$. States are arranged in increasing order such that $s_1 > \cdots > s_n$. State-space $\mathcal{S}$ can be separated into three subsets $\mathcal{S}_\downarrow$, $\mathcal{S}_0$ and $\mathcal{S}_\uparrow$, where $n_- := |\mathcal{S}_-|$, $n_0 := |\mathcal{S}_0|$, and $n_+ := |\mathcal{S}_+|$ and $n_\downarrow + n_0 + n_\uparrow = n$:

$$\begin{aligned}
\mathcal{S}_\downarrow &= \{i : s_i > R_{play}\} = \{1, \dots, n_\downarrow\},\\
\mathcal{S}_0 &= \{i : s_i = R_{play}\} = \{n_\downarrow + 1, \dots, n_\downarrow + n_0\},\\
\mathcal{S}_\uparrow &= \{i : s_i < R_{play}\} = \{n_\downarrow + n_0 + 1, \dots, n_\downarrow + n_0 + n_\uparrow\}.
\end{aligned}$$

In short, $\mathcal{S}_\downarrow$ represents the states with decreasing number of packets in flight, $\mathcal{S}_0$ represents the states with stable number of packets in flight, and $\mathcal{S}_\uparrow$ states with increasing number of packets in flight. We assume that $\varphi(t)$ can modeled such that there exists a stationary distribution π. We partition the generator matrix T as a $(n_\downarrow + n_0 + n_\uparrow) \times (n_\downarrow + n_0 + n_\uparrow)$ matrix according to:

$$T = \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} & T_{\downarrow\uparrow} \\ T_{0\downarrow} & T_{00} & T_{0\uparrow} \\ T_{\uparrow\downarrow} & T_{\uparrow 0} & T_{\uparrow\uparrow} \end{pmatrix}. \tag{2.1}$$

The combination of network congestion and play-out buffering is represented by a tandem of two fluid queues. See Figure 2.2 for an illustration of our model.

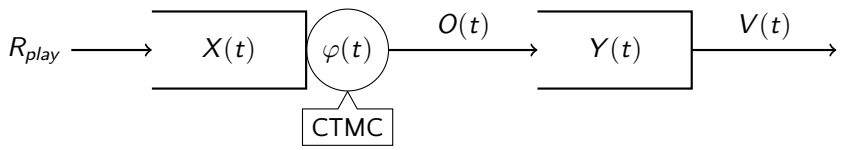


Figure 2.2: *Tandem of fluid buffers representing streaming video through an IP-network.*

The first fluid buffer models the network congestion (packets on flight), and has corresponding fluid level $X(t)$. The second fluid buffer models the play-out buffering

process at the client with corresponding fluid level $Y(t)$. Process $V(t)$ represents the video play-out rate that is achieved from the play-out buffer. For the first fluid buffer we define rates of change (of the first buffer contents) by $r_i := R_{play} - s_i$ ($i = 1, \ldots, n$), when $\varphi(t) = i$. Conversely for $\varphi(t) = i$, the rate of change in the second fluid buffer is exactly $-r_i$ whenever $Y(t) > 0$. Indeed, if $Y(t) > 0$, the play-out buffer can sustain output rate $V(t) = R_{play}$. For the second fluid buffer the rate of change is directly proportional whenever $Y(t) > 0$, so that $V(t)$ can sustain play-out at rate R_{play}. On the contrary, when $Y(t) = 0$ and $s_i < R_{play}$ the play-out buffer stays empty and $V(t) = s_i$. In this case the video stream is disturbed.

In practice the video may be stalled instead of continuously buffering and playing back. In that case the disturbed playback period in our model may be seem as a measure for the severity of distortion. We define the rate-of-change-matrix that has the same block partitioning as generator matrix T, i.e. R is a $(n_\downarrow + n_0 + n_\uparrow) \times (n_\downarrow + n_0 + n_\uparrow)$ matrix:

$$R := \begin{pmatrix} R_\downarrow & 0 & 0 \\ 0 & R_0 & 0 \\ 0 & 0 & R_\uparrow \end{pmatrix}. \tag{2.2}$$

The entries are defined by:

$$\begin{aligned} R_\downarrow &:= \text{diag}\,(r_i), && i \in S_\downarrow, \\ R_0 &:= \text{diag}\,(r_i) = 0 \quad \text{and} && i \in S_0, \\ R_\uparrow &:= \text{diag}\,(r_i), && i \in S_\uparrow. \end{aligned}$$

Here diag (r_i) $i \in S$ is the diagonal matrix with on the diagonal all elements of set S. In order for the first buffer to be stable the average potential throughput S_{res} must satisfy:

$$S_{res} := \sum_{i=1}^{n} s_i \pi_i > R_{play}. \tag{2.3}$$

The drift of the process is expressed in terms of rates of change r_i and is defined as:

$$d := \sum_{i=1}^{n} r_i \pi_i = R_{play} - S_{res}. \tag{2.4}$$

Stability condition (2.3) is equivalent to having a negative drift $d < 0$.

Due to congestion the play-out buffer level $Y(t)$ fluctuates. When the play-out buffer is empty video play-out will be disturbed as only a rate of $V(t) < R_{play}$ is supported. We consider a video stream of length $t = T_{play}$. Although we assume $S_{res} > R_{play}$ due to fluctuations in traffic the bit rate R_{play} cannot be guaranteed at all times during T_{play}. At periods with high traffic, congestion in the network builds up resulting in a temporary throughput $O(t) = s_i < R_{play}$. Therefore the video needs to be buffered at client side. When the play-out buffer is empty video play-out will be disturbed as a play-out rate of R_{play} can not be sustained. The result is that the video is quickly alternating between buffering and play-out. This is commonly experienced as being very disturbing. We want to guarantee a certain Quality of Service (QoS) on the video play-out. The QoS objective is to find an initial buffer level b_{init} such that the probability of disturbed play-out during T_{play} is smaller than p_{empty}:

$$\mathbb{P}\{\exists s \in [0, T_{play}] : V(s) < R_{play} \mid X(0) = 0, Y(0) = b_{init}\} < p_{empty}. \tag{2.5}$$

Of course the probability that play-out will be disturbed equals zero if a stream is fully buffered. However the larger the play-out buffer the longer the loading time. Second a large buffering delay causes a too large lag before the event is displayed on screen. Therefore, we want the play-out buffer to have a minimal size.

We want the play-out buffer to strike the right balance between both objectives, so that we aim for the minimal buffering threshold that guarantees undisturbed play-back with probability at least $1 - p_{empty}$. In order to minimize the initial buffer level b_{init} while meeting the QoS requirements, we develop a procedure that maps video parameters T_{play}, R_{play}, network characteristics and QoS objective p_{empty} onto a initial buffer level b_{init}.

2.3 Analysis

We are interested in a mapping from network, video characteristics and distortion probability p_{empty} to a minimal buffer level b_{init} such that Equation (2.5) is satisfied. To this end we analyze the interaction between the network congestion buffer level $X(t)$ and the play-out buffer level $Y(t)$. In our analysis four different scenarios can be identified. These are depicted in Figure 2.3. Each scenario is represented by a time interval t_i:

(1) During interval t_1 the network achieves a transfer rate lower than the video bit-rate $s_i < R_{play}$ ($r_i < 0$), while the play-out buffer level is positive $Y(t) > 0$. In this case the level of X increases while the level of Y decreases.

(2) Within interval t_2 the network transfer rate is lower than video bit-rate $s_i < R_{play}$ ($r_i < 0$), while the play-out buffer level is zero $Y(t) = 0$. Now the video playback will be disturbed and the play-out buffer level will remain zero $Y(t) = 0$ while the network content $X(t)$ continues to grow.

(3) Next, in interval t_3 we have a network transfer rate higher than the video bit-rate $s_i > R_{play}$ ($r_i > 0$), while the network content is positive $X(t) > 0$. The level of X decreases while the level of Y increases.

(4) Finally, during interval t_4 there is a network transfer rate higher than the video bit-rate $s_i > R_{play}$ ($r_i > 0$), without any backlog in the network, $X(t) = 0$. Although higher transfer rate $r_i > 0$ is supported, an effective rate of R_{play} will be achieved as the fluid entering X directly flows to the play-out buffer Y.

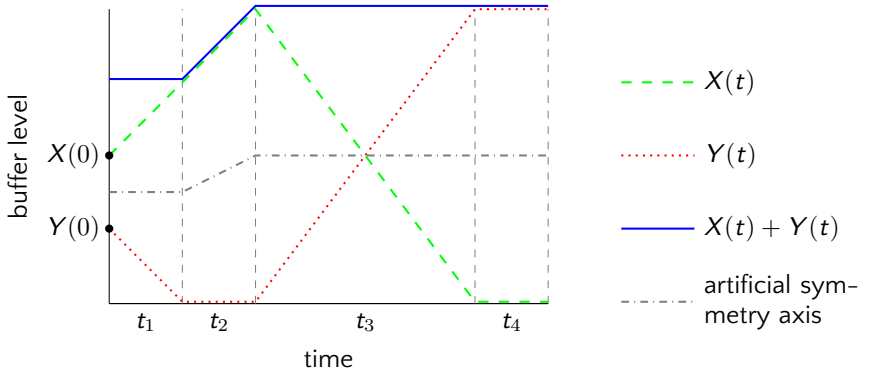


Figure 2.3: *Different phases of the stochastic processes $X(t)$ and $Y(t)$.*

Observe in Figure 2.3 that within intervals t_1, t_3 and t_4, $X(t)+Y(t)$ remains constant. Therefore, in these cases an artificial symmetry axis can be drawn between $X(t)$ and $Y(t)$. Moreover, within these intervals $V(t) = R_{play}$ and the CTMC determines how the constant level $X(t) + Y(t)$ is distributed over the first and second fluid buffer. In scenario 2 (corresponding to t_2 in Figure 2.3) the second buffer remains empty ($Y(t) = 0$) while the first buffer continues to grow. In that case $X(t)$ attains a new maximum, and obviously $X(t) = X(t) + Y(t)$ since $Y(t) = 0$. Each time $X(t)$ attains a new maximum, $X(t) + Y(t)$ grows. We can conclude that the total fluid buffer contents $X(t) + Y(t)$ is not a stationary process. However the growth of the maximum becomes an increasingly rare event each time a new maximum level is reached.

2.3.1. Definition. We define the maximum level process as

$$M^*(t) := \sup_{0 \le s \le t} X(s). \tag{2.6}$$

2.3.2. Lemma. *Let $\big(X(t), Y(t)\big)$ be the stochastic process describing fluid levels in the tandem system. Then, if $Y(0) = 0$,*

$$X(t) + Y(t) = \sup_{0 \le s \le t} X(s) = M^*(t). \tag{2.7}$$

Proof. Obviously, the initial conditions ensure that $M^*(0) = X(0) + Y(0)$. We will show that the maximum and the sum remain equal throughout time, because the maximum can only increase when $Y(t) = 0$. From the construction it is clear that, unless $Y(t) = 0$ and $\varphi(t) \in \mathcal{S}_+$, the total amount of fluid in $X(t) + Y(t)$ remains equal. Only the partition of fluid over $X(t)$ and $Y(t)$ changes as the rates of change for both buffers only differ in sign. On the contrary, when $Y(t) = 0$ and $\varphi(t) \in \mathcal{S}_+$ the amount of fluid in $X(t)$ will grow while the the second buffer remains $Y(t) = 0$ (because the inflow into the second buffer is below R_{play}). Beyond this point, both the maximum level $M^*(t)$ for $X(t)$ and $X(t)$ itself increase, as long as $Y(t)$ remains empty. We can conclude that the total amount $X(t) + Y(t)$ must always be equal to the maximum level $M^*(t)$. ∎

In Equation (2.5) we use an initial buffer level of $Y(0) = b_{init}$, while in Lemma 2.3.2 we assume $Y(0) = 0$. However, setting $Y(0) = b_{init}$ and $X(0) = 0$ corresponds to the case where $X(0)$ has a virtual (initial) supremum equal to b_{init}. Thus we are interested in the probability that new supremum $M^*(t) > b_{init}$ is attained in time interval $[0, t]$ given that the initial supremum level is set to $M^*(0) = b_{init}$. Using the connection of the initial buffer level b_{init} to the supremum level $M^*(t)$ and Lemma 2.3.2 we can rewrite Equation (2.5) to:

$$\mathbb{P}\{M^*(t) > b_{init}\} < p_{empty}. \tag{2.8}$$

This corresponds to the probability that $M^*(t)$ exceeds b_{init} when no initial-buffering is applied. We assume here and throughout the remainder of this chapter the initial condition to be $X(0) = Y(0) = 0$.

Lemma 2.3.2 targets our problem on identifying the maximum level of packets on flight. Therefore we consider the process $X(t)$. The process is driven by a CTMC and the process has negative drift. This results in a behaviour where semi regenerative busy cycles are formed each consisting of a busy period with $X(t) > 0$ that is followed by an idle period.

2.3.1 Maximum over busy cycles

In Sericola and Remiche [101] the distribution of the maximum level reached in a busy period is derived using matrix exponential forms. The resulting equations are

rewritten such that they can be transformed into matrix differential Riccati equations. Recall that the state space $\mathcal{S}$ is be partitioned into:

$$\begin{aligned}
&\mathcal{S}_{\downarrow} := \{1, \dots, n_{\downarrow}\},\\
&\mathcal{S}_0 := \{n_{\downarrow}+1, \dots, n_{\downarrow}+n_0\} \quad \text{and}\\
&\mathcal{S}_{\uparrow} := \{n_{\downarrow}+n_0+1, \dots, n_{\downarrow}+n_0+n_{\uparrow}\},
\end{aligned}$$

with corresponding rate matrices

$$\begin{aligned}
R_{\downarrow} &:= \text{diag}\,(r_i), && i \in \mathcal{S}_{\downarrow},\\
R_0 &:= \text{diag}\,(r_i) = 0 \quad \text{and}, && i \in \mathcal{S}_0,\\
R_{\uparrow} &:= \text{diag}\,(r_i), && i \in \mathcal{S}_{\uparrow},
\end{aligned}$$

that contain rates that are negative, zero or positive, respectively. For calculation of the distribution of the maximum level in a busy period, only the rates that change the buffer level ($r_i, i \notin S_0$) contribute to the solution. Moreover time is not considered in the distribution of the maximum level in a busy period. Therefore the rates can be uniformized resulting in modified matrix Q:

$$Q = \begin{pmatrix} Q_{\downarrow\downarrow} & Q_{\downarrow\uparrow} \\ Q_{\uparrow\downarrow} & Q_{\uparrow\uparrow} \end{pmatrix},$$

where the entries are defined by:

$$\begin{aligned}
Q_{\downarrow\downarrow} &= R_{\downarrow}^{-1}(T_{\downarrow\downarrow} - T_{\downarrow 0}\,T_{00}^{-1}\,T_{0\downarrow}),\\
Q_{\downarrow\uparrow} &= R_{\downarrow}^{-1}(T_{\downarrow\uparrow} - T_{\downarrow 0}\,T_{00}^{-1}\,T_{0\uparrow}),\\
Q_{\uparrow\downarrow} &= R_{\uparrow}^{-1}(T_{\uparrow\downarrow} - T_{\uparrow 0}\,T_{00}^{-1}\,T_{0\downarrow}),\\
Q_{\uparrow\uparrow} &= R_{\uparrow}^{-1}(T_{\uparrow\uparrow} - T_{\uparrow 0}\,T_{00}^{-1}\,T_{0\uparrow}).
\end{aligned}$$

2.3.3. Definition. With $\Psi_{i,j}(x)$ we define the joint distribution for the maximum level in a busy period M_+, given that a busy period starts in state $\varphi(0) = i, (i \in \mathcal{S}_{\uparrow})$ at level $X(0)$=0 and finishes in state $\varphi(\tau_0) = j, (j \in \mathcal{S}_{\downarrow})$:

$$\begin{aligned}
\Psi_{i,j}(x) &:= \mathbb{P}\big\{\varphi(\tau_0) = j, M_+ \le x \mid \varphi(0) = i, X(0) = 0\big\},\ i \in \mathcal{S}_{\uparrow}, j \in \mathcal{S}_{\downarrow}, \qquad (2.9)\\
\tau_0 &:= \inf\{t > 0 : X(t) = 0\},\\
M_+ &:= M^*(\tau_0).
\end{aligned}$$

The joint distribution of the maximum in a busy period $\Psi_{i,j}(x)$ is calculated by solving a matrix differential equation [101]. Function $\Psi_{i,j}(x)$ can be expressed in terms of the matrix exponential form of matrix Q:

$$e^{Qx} = \exp\left[\begin{pmatrix} Q_{\downarrow\downarrow} & Q_{\downarrow\uparrow} \\ Q_{\uparrow\downarrow} & Q_{\uparrow\uparrow} \end{pmatrix}\right] = \begin{pmatrix} A(x) & B(x) \\ C(x) & D(x) \end{pmatrix}. \tag{2.10}$$

The expression for $\Psi(x)$ is given by:

$$\Psi(x) = C(x)A(x)^{-1}. \tag{2.11}$$

In general we are interested in the distribution of the busy cycle $\beta(x)$ which we describe in Definition 2.3.5 below. First we introduce some further notation.

2.3.4. Definition. Matrix U is the transition matrix from an empty system to the start of a new busy cycle and is defined by:

$$\begin{aligned} U_{i,j} :=& \mathbb{P}\{\varphi(\tau_{\mathcal{S}_\uparrow}) = j \mid \varphi(0) = i,\ X(0) = 0\}, \qquad & i \in \mathcal{S}_\downarrow, j \in \mathcal{S}_\uparrow, \\ \tau_{\mathcal{S}_\uparrow} :=& \inf\{t > 0 : \varphi(t) \in \mathcal{S}_\uparrow\}, \end{aligned}$$

2.3.5. Definition. We define $\beta_{i,j}(x)$ as the joint distribution for M_+, the maximum level in a busy cycle, given that a busy period starts in state $\varphi(0) = i$ ($i \in \mathcal{S}_\uparrow$) at level $X(0)$=0 and finishes in state $\varphi(\tau_{0\uparrow}) = j$ ($j \in \mathcal{S}_\uparrow$):

$$\begin{aligned} \beta_{i,j}(x) &:= \mathbb{P}\big\{\varphi\big(\tau_{0\uparrow}\big) = j, M_+ \le x \mid \varphi(0) = i, X(0) = 0\big\}, i \in \mathcal{S}_\uparrow, j \in \mathcal{S}_\uparrow, \\ \tau_{0\uparrow} &:= \inf\{t > \tau_0 : \varphi(t) \in \mathcal{S}_\uparrow\}, \\ \tau_0 &:= \inf\{t > 0 : X(t) = 0\}. \end{aligned} \tag{2.12}$$

2.3.6. Observation. The function $\beta(x)$ can be written as $\beta(x) = \Psi(x)U$ where $\Psi(x)$ is the joint stationary distribution of the maximum level in a busy period from Definition 2.3.3. Matrix U is the transition matrix from start of an idle period to start of a busy period from Definition 2.3.4 and is given by (see for example [86, Example 1.4.4]):

$$U = -\begin{pmatrix} I & 0 \end{pmatrix}\begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0}, \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1}\begin{pmatrix} T_{\downarrow\uparrow} \\ T_{0\uparrow} \end{pmatrix}. \tag{2.13}$$

In this chapter we start with the analysis for the case where $n_\downarrow = n_\uparrow = 1$ and $n_0 = 0$. In Chapter 3 we extend the analysis for the case where $n_\downarrow \ge 1$, $n_\uparrow \ge 1$ and $n_0 \ge 0$, In general, the maxima in consecutive busy periods are not independent,

because the starting states of the environment may induce correlation. However, for the two-state model with $n_\downarrow = n_\uparrow = 1$ we have $U = [1]$. Therefore busy cycles constitute regenerative sequences, implying that maxima in consecutive busy periods *are independent*. The non-regenerative nature of the general case implies several technical complications that, while we can handle them largely analogously using semi-regenerative processes, the technical details are not part of the scope of this chapter. Instead, we specialize only for the two-state model and refer to future work for details on extensions to the semi-regenerative case.

For the two-state model with transmission rates $s_1 > R_{play}$ and $s_2 < R_{play}$, we use generator matrix:

$$T = \begin{bmatrix} -\alpha_1 & \alpha_1 \\ \alpha_2 & -\alpha_2 \end{bmatrix}$$

and rate matrix:

$$R = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}$$

to obtain the generator matrix with uniformized fluid rates:

$$Q = \begin{bmatrix} \frac{-\alpha_1}{r_1} & \frac{\alpha_1}{r_1} \\ \frac{\alpha_2}{r_2} & -\frac{\alpha_2}{r_2} \end{bmatrix}.$$

The solution the the differential equation is given by:

$$\Psi(x) = 1 - \frac{r_2\alpha_1 + r_1\alpha_2}{r_2\alpha_1 + r_1\alpha_2 e^{x(\frac{\alpha_1}{r_1} + \frac{\alpha_2}{r_2})}}.$$

The maximum of a busy cycle is given by:

$$\mathbb{P}\{M_+ \leq x\} = \Psi(x), \tag{2.14}$$

where M_+ represents the r variable corresponding to the maximum in a busy cycle. The distribution of the maximum of a busy period for the two-state model has an exponential decaying tail, and when $x \to \infty$:

$$1 - \Psi(x) = \frac{r_2\alpha_1 + r_1\alpha_2}{r_2\alpha_1 + r_1\alpha_2 e^{x(\frac{\alpha_1}{r_1} + \frac{\alpha_2}{r_2})}} \sim \left(\frac{r_1\alpha_2 + r_2\alpha_1}{r_1\alpha_2}\right) e^{-x(\frac{\alpha_1}{r_1} + \frac{\alpha_2}{r_2})}. \tag{2.15}$$

Similar to Iglehart [65, Lemma 1] we obtain an expression for

$$\mathbb{P}\{M_+ > x\} \sim be^{-\kappa x},\ x \to \infty. \tag{2.16}$$

In our case, $b = \left(\frac{r_1\alpha_2 + r_2\alpha_1}{r_1\alpha_2}\right)$ and $\kappa = (\frac{\alpha_1}{r_1} + \frac{\alpha_2}{r_2})$.

Let $M_+(k)$ be the maximum of the kth busy cycle. Using similar arguments as in Iglehart [65, Lemma 2] we obtain:

$$\lim_{n\to\infty} \mathbb{P}\{\kappa \max_{1\le k\le n} M_+(k) - \log(bn) \le x\} = \Lambda(x), \tag{2.17}$$

where

$$\Lambda(x) = \exp[-e^{-x}]. \tag{2.18}$$

Here, we use the following extreme value theorem argument:

$$\begin{aligned}
&\mathbb{P}\{\max_{1\le k\le n} M_+(k) \le \frac{x + \log(bn)}{\kappa}\} \\
&= \mathbb{P}^n\{M_+(1) \le \frac{x + \log(bn)}{\kappa}\} \\
&= \Big[1 - b\exp[-(x + \log(x + bn))] + o(\exp[-(x + \log(n))])\Big]^n.
\end{aligned}$$

2.3.2 Maximum with respect to time

Rather than the asymptotics for the busy cycles, we are interested in the evolution of the maximum over time. For this we use a result in Kulkarni and Tzenova [75]. In this chapter an expression is derived for the joint mean first passage time in a Markov Modulated fluid queue:

$$\begin{aligned}
&\mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = x, \varphi(0) = i], && i \in \mathcal{S}, \\
&\tau_{\mathcal{S}_\downarrow} := \inf\{t > 0 : X(t) = 0, \varphi(t) \in \mathcal{S}_\downarrow\}.
\end{aligned} \tag{2.19}$$

The joint mean first passage time will be represented by the function $f_i(x)$:

$$f_i(x) := \mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = x, \varphi(0) = i], \qquad i \in \mathcal{S}. \tag{2.20}$$

An expression for the joint mean first passage time can be obtained by solving the corresponding system of differential equations

$$R\frac{\mathrm{d}f(x)}{\mathrm{d}x} + Tf(x) + \overline{e} = 0, \tag{2.21}$$

with boundary condition

$$f_i(x) = 0, \qquad \forall i \in \mathcal{S}_{\downarrow}, \tag{2.22}$$

where $R = \text{diag}\,(r_1, \ldots, r_n)$ is the diagonal matrix of rates of change, T is the generating matrix and where $\overline{e}$ is a column vector of ones. Here eigenvalues λ_j are the solution to

$$\det[R - \lambda T] = 0, \tag{2.23}$$

and the corresponding right eigenvectors ϕ_j^r satisfy:

$$\lambda_i R\phi_j^r = T\phi_j^r. \tag{2.24}$$

The eigenvalues of Q, ordered such that the real parts are in increasing order:

$$\Re(\lambda_1) \leq \Re(\lambda_2) \leq \cdots \leq \Re(\lambda_{n_\uparrow}) < 0 < \Re(\lambda_{n_\uparrow+2}) \leq \cdots \leq \Re(\lambda_{n_\uparrow+n_\downarrow})$$

There are n solutions to Equation (2.23) of which there are $n_\downarrow - 1$ eigenvalues with positive real part, one eigenvalue has real part equal to 0 and there are $n_\uparrow$ eigenvalues with negative real part. In Kulkarni and Tzenova [75, Theorem 4.2] the solution for (2.21) is given by:

$$f(x) = \sum_{j=n_\uparrow+1}^{n_\uparrow+n_\downarrow} a_j\phi_j^r e^{-\lambda_j x} - \frac{\overline{e}x}{d} + g. \tag{2.25}$$

In this expression g is a solution to

$$Tg = -(cR + I)\overline{e}. \tag{2.26}$$

We are interested in the solution for the two-state model where $n_\downarrow = n_\uparrow = 1$. Plugging in T and R into the results of Kulkarni [74, Example 1] gives:

$$
\begin{aligned}
d &= \frac{\alpha_2 r_1 + \alpha_1 r_2}{\alpha_1 + \alpha_2}, \\
\lambda_1 &= 0, & \lambda_2 &= \frac{\alpha_2 r_1 + \alpha_1 r_2}{r_1 r_2}, \\
\phi_1 &= \left[1, 1\right]^t, & \phi_2 &= \left[-\frac{\alpha_1 r_2}{\alpha_2 r_1}, 1\right]^t, \\
g_1 &= \frac{r_2 - r_1}{\alpha_1 r_2 + \alpha_2 r_1}, & g_2 &= 0,
\end{aligned}
$$

which gives

$$
\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} 0 \\ -\dfrac{r_2 - r_1}{\delta} \end{bmatrix} + \begin{bmatrix} -\dfrac{\alpha_1 + \alpha_2}{\delta} \\ -\dfrac{\alpha_1 + \alpha_2}{\delta} \end{bmatrix} x, \tag{2.27}
$$

with

$$
\delta := r_2 \alpha_1 + r_1 \alpha_2.
$$

2.3.7. Definition. We define the conditional expected duration of a busy period and idle period by:

$$
\begin{aligned}
\mathbb{E}[C_B] &:= \mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = 0, \varphi(0) = i], && i \in \mathcal{S}_\uparrow, && (2.28) \\
\tau_{\mathcal{S}_\downarrow} &:= \inf\{t > 0 : X(t) = 0, \varphi(t) \in \mathcal{S}_\downarrow\}, \\
\mathbb{E}[C_I] &:= \mathbb{E}[\tau_{\mathcal{S}_\uparrow} \mid X(0) = 0, \varphi(0) = i], && i \in \mathcal{S}_\downarrow, && (2.29) \\
\tau_{\mathcal{S}_\uparrow} &:= \inf\{t > 0 : \varphi(t) \in \mathcal{S}_\uparrow\}.
\end{aligned}
$$

In the two-state model the only way that a busy period can be initiated is whenever $X(t) = 0$ and the state with $r_2 > 0$ is reached. The expected length of this busy period is equal to the first mean passage time:

$$
\begin{aligned}
\mathbb{E}[C_B] = f_2(0) = \mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = 0, \varphi(0) = 2] &= -\frac{r_2 - r_1}{r_2 \alpha_1 + r_1 \alpha_2}, && (2.30) \\
\tau_{\mathcal{S}_\downarrow} : &= \inf\{t > 0 : X(t) = 0, \varphi(t) = 1\}.
\end{aligned}
$$

There is only one state that can end a busy period and that is $\varphi(t) = 1$ when $X(t) = 0$. This initiates an idle period that continues until the state $\varphi(t) = 2$ with rate r_2 is

reached. The duration until the initiation of a consecutive busy period is exponentially distributed with mean $1/\alpha_1$. Therefore:

$$\mathbb{E}[C_I] = \mathbb{E}[\tau_\uparrow \mid X(0) = 0, \varphi(0) = 1] = 1/\alpha_1.$$

By combining the expected busy period with the expected idle period we obtain an expression for the total expected busy cycle:

$$\mathbb{E}[C] = \mathbb{E}[C_B] + \mathbb{E}[C_I] = -\frac{r_2 - r_1}{r_2\alpha_1 + r_1\alpha_2} + \frac{1}{\alpha_1} = \left(\frac{r_1}{\alpha_1}\right) \cdot \frac{\alpha_1 + \alpha_2}{r_2\alpha_1 + r_1\alpha_2}. \tag{2.31}$$

In Equation (2.17) we stated that the asymptotic distribution of the maximum of a sequence of busy cycles converges to an extreme value distribution. We now derive the asymptotic distribution over time.

Define $\{c(t) : t \geq 0\}$ as the counting process of busy cycles. Then $M^*(t)$ satisfies:

$$\max_{0 \leq k \leq c(t)} \{M_+(k) \leq x\} \leq M^*(t) \leq \max_{0 \leq k \leq c(t)+1} \{M_+(k) \leq x\}. \tag{2.32}$$

According to the weak law of large numbers we have:

$$\frac{c(t)}{t} \to \frac{1}{\mathbb{E}[C]}, \qquad t \to \infty. \tag{2.33}$$

Using Berman [13, Theorem 3.2] and Equation (2.17) the limiting distribution becomes:

$$\lim_{t\to\infty} \mathbb{P}\{\kappa M^*(t) - \log(bt) \leq x\} = \Lambda^{\frac{1}{\mathbb{E}[C]}}(x). \tag{2.34}$$

In Equation (2.34) the term $\frac{1}{\mathbb{E}[C]}$ from (2.33) represents the expected number of busy cycles per time unit (this corresponds to the c in Berman [13, Theorem 3.2]). The expression for the asymptotic distribution for the maximum of the two-state fluid queue

$$\mathbb{P}\{M^*(t) > b_{init}\} < p_{empty} \tag{2.35}$$

can now be expressed as:

$$\mathbb{P}\{\kappa M^*(t) - \log(bt) > x\} \approx 1 - \Lambda^{\frac{1}{\mathbb{E}[C]}}(x), \tag{2.36}$$

$$\mathbb{P}\{M^*(t) > b_{init}\} \approx 1 - \Lambda^{\frac{1}{\mathbb{E}[C]}}(\kappa b_{init} - \log(bt)), \tag{2.37}$$

whenever we have a sufficiently large b_{init} such that at least $b_{init} > \frac{\log(bt)}{\kappa}$.

Using the fact that when $t \to \infty$ the distribution of the maximum $M^*(t)$ converges to a Gumbel distribution, we can also establish the following asymptotic expectation of the maximum level:

$$\mathbb{E}[M^*(t)] \to \frac{\log\left(\frac{bt}{\mathbb{E}[C]}\right) + \gamma}{\kappa}, \qquad t \to \infty, \tag{2.38}$$

where $\gamma \approx 0.577215665$ is the Euler-Mascheroni constant. The behavior with respect to the real process is illustrated in Figure 2.6. Observe that $\mathbb{E}[M^*(t)]$ grows logarithmically over time with logarithmic slope $\frac{1}{\kappa}$.

2.4 Dimensioning the initial buffer size

In Section 2.3 we showed that the probability of an empty play-out buffer corresponds to the maximum level reached by the first fluid buffer representing the number of packets in flight. Given the parameters that capture the network behavior (s and T) for a video stream with bit-rate R_{play} and duration T_{play} the initial buffer level b_{init} can be determined. Given the video playback QoS parameter p_{empty}, that represents the maximum probability a video is disturbed during T_{play}, the initial buffer size b_{init} should be chosen such that:

$$b_{init} > \frac{-\log[-\frac{\mathbb{E}[C]}{bT_{play}}\log(1-p_{empty})]}{\kappa}. \tag{2.39}$$

This holds when we have T_{play} sufficiently large such that

$$T_{play} > -\log(1-p_{empty})\frac{\mathbb{E}[C]}{b}.$$

This is a reasonable assumption, since we are considering video streams that have typically long durations (minutes and longer) compared to the time scale of fluctuations in the network transmission speed (typically in the order of seconds).

2.5 Numerical experiments

In the previous sections we derived a mapping from the QoS parameter p_{empty} and streaming video duration T_{play} to minimal initial buffer level b_{init}. We will now run

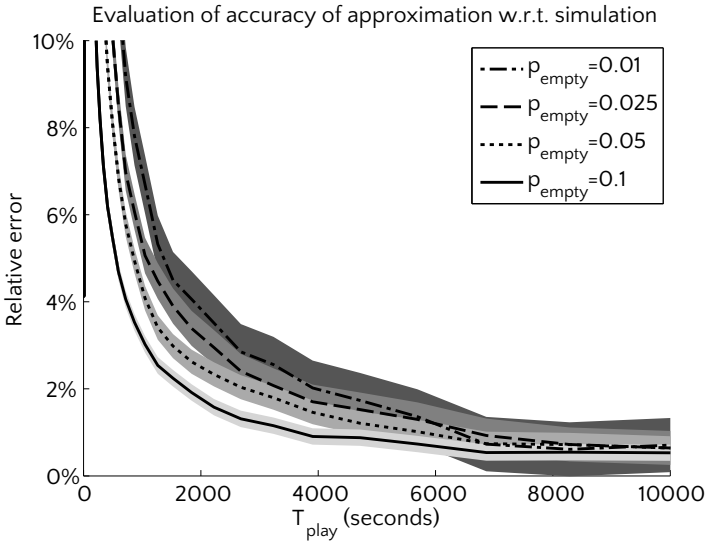


Figure 2.4: *Relative difference of buffer under-run probability to simulation. The gray bands around the lines are the 95% confidence intervals of the simulation.*

simulations in order to evaluate the accuracy of our mapping. Our parameter setting is as follows: $\alpha_1 = 0.1$, $\alpha_2 = 0.2$, $s_1 = 8$*Mbps*, $s_2 = 2$*Mbps*, $R_{play} = 4$*Mbps*, $r_1 = -4$, $r_2 = 2$, $R = \text{diag}\left(\begin{bmatrix} r_1 & r_2 \end{bmatrix}\right)$ and

$$T = \begin{bmatrix} -\alpha_1 & \alpha_1 \\ \alpha_2 & -\alpha_2 \end{bmatrix}.$$

The simulation consists of $10,000,000$ sample paths. Figure 2.4 represents the relative difference between target tail probability p_{empty} and the actual fraction of sample paths that exceed the buffer level approximation. We define the relative difference of approximation (app) and simulation (sim) by:

$$\text{diff}_{relative}(app, sim) = \left| \frac{app - sim}{sim} \right|. \tag{2.40}$$

From Figure 2.4 it can be observed that for the the tail probability $\mathbb{P}\{M(T_{play}) > b_{init}\}$ the error quickly approaches the region below 5%. Figure 2.5 represents the actual fraction of sample paths that exceeds the theoretical asymptotic percentiles. The theoretical percentiles are based on Equation (2.39). The straight thin dashed lines represent the desired tail probability.

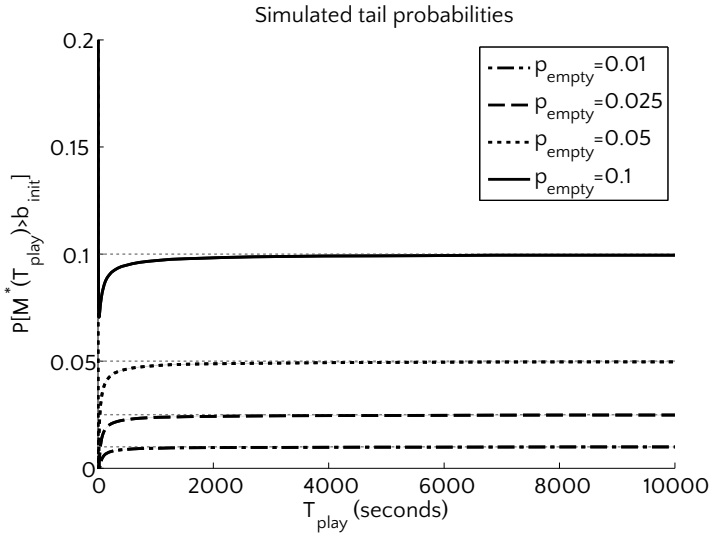


Figure 2.5: *Tail probabilities using empirical distribution based on simulation, evaluated on theoretical asymptotic percentiles.*

The tail probabilities in Figure 2.5 indicate that the buffer level, derived from asymptotics, gives a conservative estimate, i.e., an overestimation of the tail probability. So using the asymptotics, depending on the duration of the video stream, the estimated buffer level is slightly higher than strictly needed.

In Section 2.3.2 we derived the asymptotic mean in Equation (2.38). We compare the asymptotic mean to the simulation results in Figure 2.6. This figure has a logarithmic time scale because we expect the mean maximum level to asymptotically converge to logarithmic growth with respect to time. From Figure 2.6 we observe that this is indeed the case.

In Figure 2.7 percentiles from simulation are compared to the theoretical asymptotic percentiles. Black lines represent simulation percentiles while gray lines represent the theoretical percentiles as expressed in Equation (2.39). On a linear time scale, simulation and asymptotic percentiles coincide quite closely.

Figure 2.8 presents the percentiles on logarithmic time scale. On small time scale we observe a "notch" in the simulation percentiles. This is caused by the fact that the figure is presented in logarithmic time scale. From the buffer process we can derive a coarse upper bound. A percentile at time t can not exceed $t \max(R - s_i)$ as $\max(R - s_i)$ is the maximal possible growth rate of $X(t) + Y(t)$. The "notch" corresponds to the upper bound (which is curved due to logarithmic time scale).

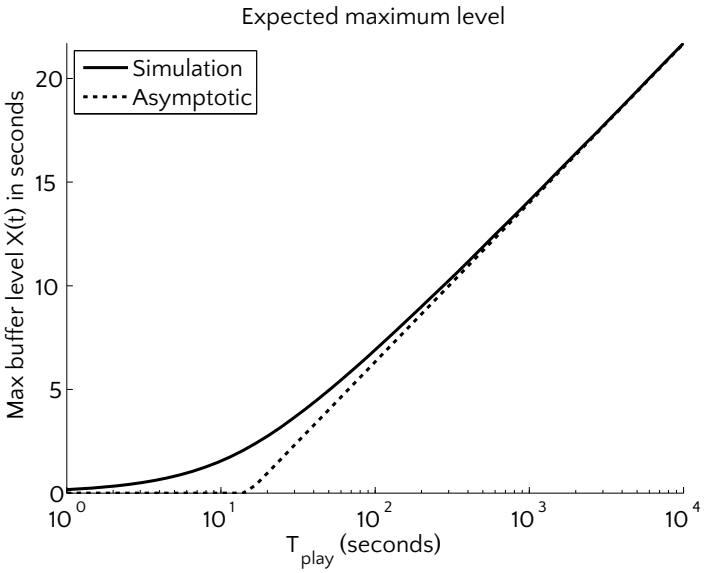


Figure 2.6: *Simulated and theoretical (asymptotic, see Equation (2.38)) expectation of the maximum level $M^*(t)$ on logarithmic time-scale.*

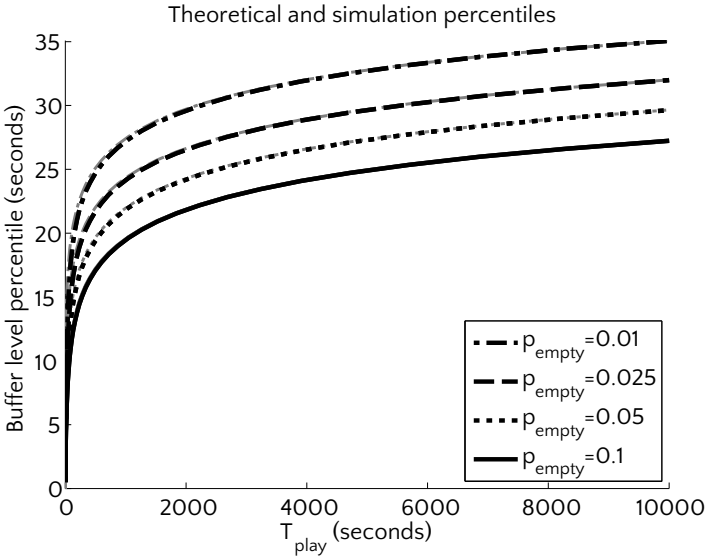


Figure 2.7: *Black lines represent simulation percentiles, gray lines represent theoretical asymptotic percentiles.*

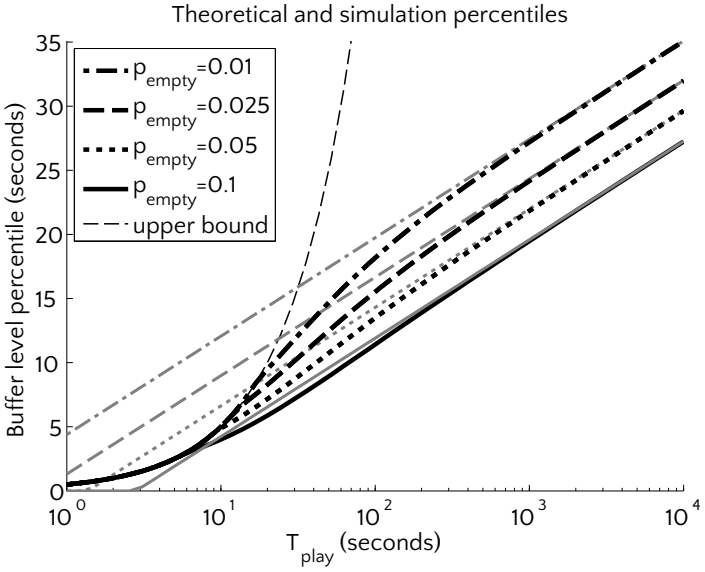


Figure 2.8: *Percentiles on logarithmic time scale. Black lines represent percentiles from simulation, gray lines represent theoretical asymptotic percentiles.*

2.6 Discussion

We studied a model for a constant bit-rate video stream over an IP network with a play-out buffer at the client side. The network is modeled as a Markov Modulated fluid queue in which a CTMC determines the actual transmission rate through the network. For the play-out buffer an initial buffer level b_{init} was determined such that the probability that the video will stall during play-out will not exceed an agreed service level probability p_{empty}.

2.6.1. Remark (Variable bitrate)**.** In our exposition, we assumed that the video application was streamed at constant bit rate. For practical application, however, it is more realistic to assume that the video produces variable bit rate flows. Our model still applies to this case, if we take the transport unit to be *time* rather than *bits* or *packets*. The streaming and play-out rate are then $R_{play} = 1$ (one unit of time is played each unit of time). To incorporate the variable bit rate into our model, we modify the network throughput process $\varphi(t)$ as follows. We construct it from two independent components $\varphi(t) = (\varphi^1(t), \varphi^2(t))$. The first component is a CTMC and again determines the network capacity at time t in *bits per time unit*, say speed s_i^1 if $\varphi^1(t) = i$. The second component $\varphi_2(t)$ is also a CTMC, independent of $\varphi_1(t)$, and determines the *length of time encoded per bit* for the video segments transported

through the network at time t, say s_j^2 if $\varphi^2(t) = j$. Setting the network speeds as $s_{i,j} = s_i^1 s_j^2$ whenever $\varphi(t) = (i, j)$, our original model can be directly used. Of course, exploiting the structure of the process $\varphi(t)$ (its generator, for example, can be written as the Kronecker product of the generators of φ^1 and φ^2) was not part of the scope of our analysis here. Incorporating this structure may further enhance efficient computations.

We have shown that the probability of this event corresponds to the event of the maximum congestion level $M(t)$ exceeding the initial buffer level b_{init}. As a by-product, we found that the asymptotic distribution of the maximum level $M(t)$, $t \to \infty$ has a Gumbel distribution, which is in agreement with earlier results in [4]. For smaller t the expression of the asymptotic distribution can be used to approximate the tail probability $\mathbb{P}\{M(T) > b_{init}\}$. From this expression we derived a formula that maps p_{empty}, T_{play} and the network and video parameters to a minimal buffer level b_{init}. Simulation results indicate that the buffer level that is obtained from the asymptotic analysis is a conservative estimate, i.e., it overestimates the true minimal required buffer level. The longer the video stream the more accurate the asymptotic prediction is. In adaptive media streaming, streaming servers tend to adapt R_{play} to the fluctuating available bandwidth. Our analysis facilitates proper parameter selection with respect to the altered network parameters.

2.6.2. Remark (Transition probabilities)**.** For practical purposes it may be difficult to estimate the transition probabilities of the modulating process $\varphi(t)$. In principle, this can be done using the classical maximum likelihood estimators as described for example in [86, Section 1.10]. For the choice of the state space it is natural to let the state of the modulating process coincide with the measured network rate; the granularity then determines the dimension of the transition matrix. In practice, one may however not want to go into estimation of the network characteristics, but rather try to adapt the coefficients κ and $\mathbb{E}[C]/b$ in the dimensioning rule formulated in relation 2.39. Through live measurements, one may decide on adapting the estimates for these coefficients so as to improve quality when the stall probability is too large, or reduce the initial delay, when the buffer is never close to empty.

A Spectral Theory Approach for Extreme Value Analysis in a Tandem of Fluid Queues

3

In Chapter 2 we studied a model for a constant bit-rate video stream over an IP network with a play-out buffer at the client side. The network is modeled as a Markov Modulated fluid queue in which a CTMC determines the actual transmission rate through the network. For the play-out buffer we derived an initial buffer level b_{init} such that the probability that the video will stall during play-out will not exceed an agreed service level probability p_{empty}. We demonstrated that the probability of this event corresponds to the event of the maximum congestion level $M(t) := \sup_{0 \leq s \leq t} X(s)$ exceeding the initial buffer level b_{init}. The analysis was focused on the case with two states.

This chapter extends the result of Chapter 2 to an arbitrary number of states. To this end we extend results on the maximum level in a busy period from Section 2.3.1, the conditional busy cycle duration, the expected busy cycle duration from and extreme value theorem asymptotic result in Section 2.3.2. Our analysis was particularly motivated by the cited papers of Berman [13] and Iglehart [65]. For a literature review we refer to Section 2.1. We further refer to Asmussen [5] for an excellent survey on extreme-value theory for queues.

For the case with more than two states there is not always an explicit expression for the initial level of the play-out-buffer. However we provide an explicit recipe to calculate the asymptotic behavior of the maximum level in the Markov Modulated fluid queue. This recipe contains results that were derived using spectral theory analysis on the fluid model equations. The result can directly be applied to dimension the initial play-out buffer size.

The organization of the remainder of this chapter is as follows. In Section 3.1 we lay out the modifications to our model and the extension to the analysis described in Sections 2.2 and 2.3. We also describe how the dimensioning rule for the initial buffer level extends to the more general case.

In Section 3.2, we provide a numerical validation of the proposed dimensioning rule by means of simulations. Section 3.3 contains a discussion of the results and looks out to future work.

[3] This chapter is based on [22] and [20].

3.1 Analysis

This section extends our analysis of Chapter 2. For the model formulation and required preliminaries for this section we refer to Sections 2.2 and 2.3.

We show that the expression for the distribution of the maximum in a busy cycle has an exponential tail. Moreover we can derive an explicit expression for the asymptotic tail. In the expression for $\Psi(x)$ from (2.11) function $C(x)$ is an $n_\uparrow \times n_\downarrow$ matrix and $A(x)$ is an $n_\downarrow \times n_\downarrow$ matrix. For the case $n_\downarrow > 1$ we have to take the inverse of a matrix that contains exponential terms with exponents corresponding to the eigenvalues of Q. Using Sylvester's formula [38, Page 87] the matrix exponential e^{Qt} can be decomposed as:

$$e^{Qx} = e^{\lambda_1 x}\widetilde{Q}_1 + \ldots + e^{\lambda_{n_\downarrow+n_\uparrow} x}\widetilde{Q}_{n_\downarrow+n_\uparrow}, \tag{3.1}$$

where the eigenvalues $\lambda_1, \ldots, \lambda_{n_\downarrow+n_\uparrow}$ of Q are the solution of

$$\det[Q - \lambda I] = 0, \tag{3.2}$$

and the matrices $\widetilde{Q}_i, \quad i = 1, \ldots, n_\downarrow + n_\uparrow$ are the Frobenius covariants. Let ϕ_i^l and ϕ_i^r be the normalised left and right eigenvector corresponding to eigenvalue λ_i:

$$\phi_i^l Q = \lambda_i \phi_i^l \quad \text{and} \tag{3.3}$$
$$Q\phi_i^r = \lambda_i \phi_i^r, \tag{3.4}$$

respectively. The corresponding Frobenius covariants are given by $\widetilde{Q}_i = \phi_i^r \phi_i^l$. These describe how the exponentials $e^{\lambda_i x}$ with corresponding eigenvalues λ_i contribute to the matrix exponential e^{Qx}. If we consider the partitioning of e^{Qx} in Equation (2.10) then $C(x)$ and $A(x)$ can be represented as:

$$A(x) = e^{\lambda_1 x}\widetilde{A}_1 + \ldots + e^{\lambda_{n_\downarrow+n_\uparrow} x}\widetilde{A}_{n_\downarrow+n_\uparrow} \quad \text{and} \tag{3.5}$$
$$C(x) = e^{\lambda_1 x}\widetilde{C}_1 + \ldots + e^{\lambda_{n_\downarrow+n_\uparrow} x}\widetilde{C}_{n_\downarrow+n_\uparrow}. \tag{3.6}$$

Now we decompose (2.11) into:

$$\Psi(x) = \frac{C(x)\,\mathrm{adj}\,[A(x)]}{\det\,[A(x)]}. \tag{3.7}$$

As $A(x)$ is $n_\downarrow \times n_\downarrow$ both the determinant of $A(x)$ and the product $C(x)\,\mathrm{adj}\left[A(x)\right]$ will contain terms that are products of $n_\downarrow$ exponentials. The resulting exponential terms have exponents that are sums of $n_\downarrow$ eigenvalues.

3.1.1. Definition. Let c be a vector with n elements. In summations we denote with

$$\sum_{k\in c} := \sum_{\substack{k=c_i,\\ i=1,\ldots,n}}$$

that we iterate k over the elements from vector $c = (c_1, \ldots, c_n)$.

3.1.2. Lemma. *Let A be an $n \times n$ matrix and $m \geq 1$:*

$$A = \sum_{k=1}^{m} b_k A_k,$$

with

$$A_k = \boldsymbol{r}_k^T \boldsymbol{c}_k = \begin{bmatrix} r_{k,1}c_{k,1} & \cdots & r_{k,1}c_{k,n} \\ \vdots & \ddots & \vdots \\ r_{k,n}c_{k,1} & \cdots & r_{k,n}c_{k,n} \end{bmatrix}.$$

Then the following holds:

$$\mathrm{adj}\,[A] = \sum_{c\in\mathcal{C}} \left(\prod_{k\in c} b_k\right) \mathrm{adj}\left[\sum_{k\in c} A_k\right],$$

where $\mathcal{C}$ is the set with all combinations of length $n-1$ from the set $\{1, 2, \ldots, m\}$.

Proof. For the proof we refer to Appendix 3.A. ∎

3.1.3. Lemma. *Let*

$$A(x) = e^{\lambda_1 x}\widetilde{A}_1 + \ldots + e^{\lambda_{n_\downarrow+n_\uparrow} x}\widetilde{A}_{n_\downarrow+n_\uparrow} \quad \textit{and}$$
$$C(x) = e^{\lambda_1 x}\widetilde{C}_1 + \ldots + e^{\lambda_{n_\downarrow+n_\uparrow} x}\widetilde{C}_{n_\downarrow+n_\uparrow},$$

with

$$\widetilde{Q}_k = \phi_k^r \phi_k^l = \begin{bmatrix} \phi_{k,1}^r \phi_{k,1}^l & \cdots & \phi_{k,1}^r \phi_{k,n_\downarrow+n_\uparrow}^l \\ \vdots & \ddots & \vdots \\ \phi_{k,n_\downarrow+n_\uparrow}^r \phi_{k,1}^l & \cdots & \phi_{k,n_\downarrow+n_\uparrow}^r \phi_{k,n_\downarrow+n_\uparrow}^l \end{bmatrix},$$
$$= \begin{bmatrix} \widetilde{A}_k & \widetilde{B}_k \\ \widetilde{C}_k & \widetilde{D}_k \end{bmatrix},$$

where $\widetilde{A}_k$ is $n_\downarrow \times n_\downarrow$, $\widetilde{B}_k$ is $n_\downarrow \times n_\uparrow$, $\widetilde{C}_k$ is $n_\uparrow \times n_\downarrow$ and $\widetilde{D}_k$ is $n_\uparrow \times n_\uparrow$. Furthermore, let $\mathcal{C}$ be the set of combinations of length n from the set $\{1, \dots, n\}$. Then the following holds:

$$C(x)\,\mathrm{adj}\,[A(x)] = \sum_{c\in\mathcal{C}} \left(\prod_{k\in c} e^{\lambda_k x} \right) \sum_{k\in c} \widetilde{C}_k\,\mathrm{adj} \left[\sum_{k\in c} \widetilde{A}_k \right]. \tag{3.8}$$

Proof. By applying Lemma 3.1.2 we obtain:

$$C(x)\,\mathrm{adj}\,[A(x)] = \sum_{j=1}^{n_\downarrow+n_\uparrow} e^{\lambda_j x} \widetilde{C}_j \sum_{c\in\overline{\mathcal{C}}} \left(\prod_{k\in c} e^{\lambda_k x} \right) \mathrm{adj} \left[\sum_{k\in c} \widetilde{A}_k \right],$$

with $\overline{\mathcal{C}}$ the set of combinations of $n_\downarrow - 1$ elements from the set $\{1, \dots, n_\downarrow + n_\uparrow\}$. From (2.10) and Observation 2.3.6 we find that both $\widetilde{A}_k$ and $\widetilde{C}_k$ share the same row vector. As all sums of $n_\downarrow - 1$ matrices $\widetilde{A}_k$ have rank $n_\downarrow - 1$ the following is true:

$$\sum_{k\in c} \widetilde{C}_k\,\mathrm{adj} \left[\sum_{k\in c} \widetilde{A}_k \right] = 0, \qquad \forall c \in \overline{\mathcal{C}}. \tag{3.9}$$

Using (3.9) we can rewrite:

$$\begin{aligned} C(x)\,\mathrm{adj}\,[A(x)] &= \sum_{j=1}^{n_\downarrow+n_\uparrow} e^{\lambda_j x}\widetilde{C}_j \sum_{c\in\overline{\mathcal{C}}}\left(\prod_{k\in c} e^{\lambda_k x}\right)\mathrm{adj}\left[\sum_{k\in c}\widetilde{A}_k\right], \\ &= \sum_{c\in\overline{\mathcal{C}}}\sum_{j\notin c} e^{\lambda_j x}\widetilde{C}_j\left(\prod_{k\in c} e^{\lambda_k x}\right)\mathrm{adj}\left[\sum_{k\in c}\widetilde{A}_k\right], \\ &= \sum_{c\in\mathcal{C}}\sum_{j\in c} e^{\lambda_j x}\widetilde{C}_j\left(\prod_{k\in c\setminus j} e^{\lambda_k x}\right)\mathrm{adj}\left[\sum_{k\in c\setminus j}\widetilde{A}_k\right], \\ &= \sum_{c\in\mathcal{C}}\left(\prod_{k\in c} e^{\lambda_k x}\right)\sum_{j\in c}\widetilde{C}_j\,\mathrm{adj}\left[\sum_{k\in c\setminus j}\widetilde{A}_k\right]. \end{aligned}$$

By using again (3.9) we obtain:

$$\begin{aligned} &\sum_{c\in\mathcal{C}}\left(\prod_{k\in c} e^{\lambda_k x}\right)\sum_{j\in c}\widetilde{C}_j\,\mathrm{adj}\left[\sum_{k\in c\setminus j}\widetilde{A}_k\right], \\ &= \sum_{c\in\mathcal{C}}\left(\prod_{k\in c} e^{\lambda_k x}\right)\sum_{k\in c}\widetilde{C}_k\,\mathrm{adj}\left[\sum_{k\in c}\widetilde{A}_k\right]. \end{aligned}$$

■

3.1.4. Observation. There are $m = \binom{n_\downarrow+n_\uparrow}{n_\downarrow}$ unique combinations of $n_\downarrow$ eigenvalues from $n_\downarrow+n_\uparrow$ eigenvalues. Let $c\in\mathcal{C}$ be the set of combinations of $n_\downarrow$ indices from the set $\{1,2,\dots,n_\downarrow+n_\uparrow\}$. We define the sums of eigenvalues $\lambda_{k_1},\dots,\lambda_{k_{n_\downarrow}}$ corresponding to combination c_k with index k by:

$$\widehat{\lambda}_k := \sum_{j\in c_k}\lambda_j, \quad k=1,\dots,m, \quad c_k\in\mathcal{C},$$

where the set $\mathcal{C}$ is ordered in decreasing order according to the real parts of $\hat{\lambda}_k$ such that:

$$\Re\big(\widehat{\lambda}_1\big)\geq\Re\big(\widehat{\lambda}_2\big)\geq\ldots\geq\Re\big(\widehat{\lambda}_m\big).$$

3.1.5. Lemma. *Equation* (3.7) *can be rewritten as:*

$$\Psi(x)=\frac{\widehat{C}_1e^{\widehat{\lambda}_1}+\widehat{C}_2e^{\widehat{\lambda}_2}+\ldots+\widehat{C}_me^{\widehat{\lambda}_m}}{\widehat{A}_1e^{\widehat{\lambda}_1}+\widehat{A}_2e^{\widehat{\lambda}_2}+\ldots+\widehat{A}_me^{\widehat{\lambda}_m}}, \tag{3.10}$$

with values $\hat{\lambda}_k$ as defined in Observation 3.1.4, and

$$\widehat{C}_k := \sum_{j \in c_k} \widetilde{C}_j \operatorname{adj}\Big[\sum_{j \in c_k} \widetilde{A}_j\Big], \qquad c_k \in \mathcal{C},$$

and

$$\widehat{A}_k := \det\Big[\sum_{j \in c_k} \widetilde{A}_j\Big], \qquad c_k \in \mathcal{C},$$

where the elements c_k from set $\mathcal{C}$ are ordered according to Observation 3.1.4 such that:

$$\Re(\widehat{\lambda}_1) \geq \Re(\widehat{\lambda}_2) \geq \ldots \geq \Re(\widehat{\lambda}_m).$$

Proof. Due to the determinant and adjoint matrix in Equation (3.7), there will be exponential terms in both numerator and denominator that result from products of $n_\downarrow$ exponentials $e^{\lambda_i x}$ with eigenvalues $\lambda_i,\ i \in \mathcal{S}$. First consider the terms in the denominator. Remember that the Frobenius covariants $\widetilde{Q}_i$ (and also $\widetilde{A}_i$, $\widetilde{C}_i$) have rank 1. Therefore only linear combinations of $n_\downarrow$ distinct Frobenius covariants, defined by $c_k \in \mathcal{C}$, will result in positive determinants. Combination $c_k \in \mathcal{C}$ is element of the set containing al combinations of length $n_\downarrow$ from the set $\{1, \ldots, n_\downarrow + n_\uparrow\}$ as defined in Observation 3.1.4. Considering the numerator, the adjoint matrix of a linear combination of Frobenius covariants $\widetilde{A}_i$ will only have positive entries when it is a linear combination of $n_\downarrow - 1$ distinct Frobenius covariants as the adjoint matrix contains minors of degree $n_\downarrow - 1$. By applying Lemma 3.1.3 we observe that only remaining exponential terms in the numerator are those that correspond to sums over combinations $c_k \in \mathcal{C}$ of $n_\downarrow$ eigenvalues. ∎

As $\widehat{\lambda}_k$ is ordered in decreasing order the leading exponential term is $e^{\widehat{\lambda}_1}$. Considering (3.7) the limiting distribution Ψ^∞ becomes:

$$\Psi^\infty := \lim_{x \to \infty} \Psi(x) = \frac{\widehat{C}_1}{\widehat{A}_1}. \tag{3.11}$$

Using this we can derive the tail behaviour of $\Psi(x)$:

3.1.6. Lemma. *$\Psi(x)$ has an exponential tail that behaves as*

$$\Psi^\infty - \Psi(x) \to G e^{-\kappa x}, \qquad x \to \infty, \tag{3.12}$$

where

$$\kappa = \lambda_{n_\uparrow}, \quad G = \frac{\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2}$$

and κ is the maximal (least) negative eigenvalue of Q.

Proof. Subtracting Ψ^∞ from the expression of $\Psi(x)$ in Lemma 3.1.5 gives:

$$\begin{aligned}\Psi^\infty - \Psi(x) &= \frac{\widehat{C}_1}{\widehat{A}_1} - \frac{\widehat{C}_1 e^{\widehat{\lambda}_1} + \widehat{C}_2 e^{\widehat{\lambda}_2} + \ldots + \widehat{C}_m e^{\widehat{\lambda}_m}}{\widehat{A}_1 e^{\widehat{\lambda}_1} + \widehat{A}_2 e^{\widehat{\lambda}_2} + \ldots + \widehat{A}_m e^{\widehat{\lambda}_m}}, \\ &= \frac{[\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2]e^{\widehat{\lambda}_2} + \ldots + [\widehat{C}_1\widehat{A}_m - \widehat{A}_1\widehat{C}_m]e^{\widehat{\lambda}_m}}{\widehat{A}_1[\widehat{A}_1 e^{\widehat{\lambda}_1} + \widehat{A}_2 e^{\widehat{\lambda}_2} + \ldots + \widehat{A}_m e^{\widehat{\lambda}_m}]}.\end{aligned}$$

When $x \to \infty$ the two leading exponential terms $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$ remain:

$$\Psi^\infty - \Psi(x) \to \frac{\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2} e^{\widehat{\lambda}_2 - \widehat{\lambda}_1}, \qquad x \to \infty. \tag{3.13}$$

According to Kulkarni [74, Theorem 11.5] the eigenvalues of Q, resulting from $\det[R - \lambda T] = 0$ can be ordered as follows:

$$\Re(\lambda_1) \leq \Re(\lambda_2) \leq \ldots \leq \Re(\lambda_{n_\uparrow}) < 0 < \Re(\lambda_{n_\uparrow+2}) \leq \cdots \leq \Re(\lambda_{n_\uparrow+n_\downarrow}). \tag{3.14}$$

there are $n_\uparrow$ eigenvalues with negative real part, one eigenvalue is equal to zero and there are $n_\downarrow - 1$ eigenvalues with positive real part. In Definition 3.1.4 we defined $\widehat{\lambda}_k$ as the sum of $n_\downarrow$ unique eigenvalues. Consider $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$:

$$\begin{aligned}\widehat{\lambda}_1 &= 0 \quad + \lambda_{n_\uparrow+2} + \ldots + \lambda_{n_\uparrow+n_\downarrow}, \\ \widehat{\lambda}_2 &= \lambda_{n_\uparrow} + \lambda_{n_\uparrow+2} + \ldots + \lambda_{n_\uparrow+n_\downarrow}.\end{aligned}$$

Observe that $\widehat{\lambda}_1$ consists of $n_\downarrow - 1$ eigenvalues with positive real part and one eigenvalue equal to zero. The next $\widehat{\lambda}_2$ is obtained by replacing the eigenvalues equal to zero with the eigenvalue with least negative real part $\lambda_{n_\uparrow}$. Therefore

$$\widehat{\lambda}_2 - \widehat{\lambda}_1 = \max_{i \in \{i:\lambda_i<0\}} \lambda_i = \lambda_{n_\uparrow}.$$

Plugging this in (3.13) gives:

$$\Psi^\infty - \Psi(x) \to G e^{\kappa x}, \qquad x \to \infty,$$

with

$$G := \frac{\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2}$$

and

$$\kappa := \max_{i\in\{i:\lambda_i<0\}} \lambda_i = \lambda_{n_\uparrow}.$$

■

From Lemma 3.1.6 we established that $\Psi(x)$ has an exponential tail $Ge^{-\kappa x}$. Here G is a matrix while we are interested in the general case averaging over all transitions. Therefore we define the following transition matrices:

3.1.7. Definition.

$$P_{BI} := \Psi^\infty = \frac{\widehat{C}_1}{\widehat{A}_1}, \tag{3.15}$$

$$P_{IB} := U = -\begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1} \begin{pmatrix} T_{\downarrow\uparrow} \\ T_{0\uparrow} \end{pmatrix}, \tag{3.16}$$

$$P_{BB} := P_{BI}P_{IB} = -\begin{pmatrix} \Psi^\infty & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1} \begin{pmatrix} T_{\downarrow\uparrow} \\ T_{0\uparrow} \end{pmatrix}, \tag{3.17}$$

where P_{BI} is the transition matrix from a busy to an idle period, P_{IB} is the transition matrix from an idle to a busy period and P_{BB} is the transition matrix between states that initiate busy cycles. In P_{BI}, Ψ^∞ is transition matrix from a state that initiates a busy period to the state that terminates the busy period. Recall that U is the transition matrix from Definition 2.3.4 for transitions from idle period states to busy period initiating states.

We use transition matrix P_{BB} for calculating the stationary distribution π_B over states $(i \in \mathcal{S}_\uparrow)$ that initiate a busy period. The stationary distribution π_B is the solution of:

$$\begin{aligned} &\pi_B P_{BB} = \pi_B, \\ &\sum \pi_B = 1. \end{aligned} \tag{3.18}$$

3.1.8. Corollary. *The overall expected tail of the distribution on the maximum is given by:*

$$\mathbb{P}\{M_+ > z\} \to be^{-\kappa z}, \qquad x \to \infty, \tag{3.19}$$

where $b = \pi_B \frac{\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2}\overline{e}$ *and* $\kappa = \lambda_{n_\uparrow}$.

Proof. The stationary distribution of states that initiate a busy period is given by π_B. The marginal distribution of the maximum in a busy period is given by $\Psi(x)$ and is conditioned on the states $i \in \mathcal{S}_\uparrow$ that initiate a busy period. The overall distribution of the maximum is given by:

$$\pi_B \Psi(x)\overline{e}.$$

We have to add the rows and weight the sums according to the stationary distribution π_B. The same holds for the exponential tail parameter G from Lemma 3.1.6:

$$b := \pi_B G\overline{e}. \tag{3.20}$$

We define the maximum of an arbitrary busy cycle by:

$$\mathbb{P}\{M_+ \leq x\},$$

where M_+ represents the stochastic variable corresponding to the maximum of the busy cycle. Similar to Iglehart [65, Lemma 1] we obtain an expression for

$$\mathbb{P}\{M_+ > z\} \to be^{-\kappa z}, \qquad x \to \infty.$$

In our case $b = \pi_B \frac{\widehat{C}_1\widehat{A}_2 - \widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2}\overline{e}$ and $\kappa = \lambda_{n_\uparrow}$. ∎

3.1.9. Lemma. *Let* $M_+(k)$ *be the maximum of the kth busy cycle. Then the following holds:*

$$\lim_{n\to\infty} \mathbb{P}\{\kappa \max_{1\leq k\leq n} M_+(k) - \log(bn) \leq x\} = \Lambda(x), \tag{3.21}$$

where

$$\Lambda(x) = \exp[-e^{-x}]. \tag{3.22}$$

Proof. In Corollary 3.1.8 we showed that the maximum of a busy cycle has an exponential tail according to:

$$\mathbb{P}\{M_+ > z\} \to be^{-\kappa z}, \qquad x \to \infty.$$

Using the same arguments as in Iglehart [65, Lemma 2] we can derive that

$$\lim_{n\to\infty} \mathbb{P}\{\kappa \max_{1\le k\le n} M_+(k) - \log(bn) \le x\} = \Lambda(x).$$

The following extreme value theorem argument can be used:

$$\begin{aligned}&\mathbb{P}\{\max_{1\le k\le n} M_+(k) \le \frac{x+\log(bn)}{\kappa}\} = \mathbb{P}^n\{M_+(1) \le \frac{x+\log(bn)}{\kappa}\},\\ &= \left[1 - b\exp[-(x+\log(x+bn))] + o(\exp[-(x+\log(n))])\right]^n.\end{aligned}$$

■

3.1.1 Maximum with respect to time

Rather than the asymptotics for the busy cycles, we are interested in the evolution of the maximum over time.

For this we use a result in Kulkarni and Tzenova [75], who derive an expression for the joint mean first passage time in a Markov Modulated fluid queue:

$$\begin{aligned}&\mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = x, \varphi(0) = i], && i \in \mathcal{S},\\ &\tau_{\mathcal{S}_\downarrow} := \inf\{t > 0 : X(t) = 0, \varphi(t) \in \mathcal{S}_\downarrow\}.\end{aligned}$$

The joint mean first passage time will be represented by function $f_i(x)$:

$$f_i(x) := \mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = x, \varphi(0) = i]. \qquad i \in \mathcal{S}. \tag{3.23}$$

An expression for the joint mean first passage time can be obtained by solving a system of differential equations:

$$R\frac{\mathrm{d}f(x)}{\mathrm{d}x} + Tf(x) + \overline{e} = 0. \tag{3.24}$$

with boundary condition:

$$f_i(x) = 0, \qquad \forall i \in \mathcal{S}_\downarrow. \tag{3.25}$$

where $R = \text{diag}(r_1, \dots, r_n)$ is the diagonal matrix with rates of change, T is the generating matrix and where $\overline{e}$ is a column vector of ones. Here eigenvalues λ_j as the solution to

$$\det[R - \lambda T] = 0, \tag{3.26}$$

and corresponding right eigenvectors ϕ_j^r for which holds:

$$\lambda_i R \phi_j^r = T \phi_j^r. \tag{3.27}$$

Note that the eigenvalues are equal to the eigenvalues obtained in (3.2). Recall that the eigenvalues of Q, are ordered in increasing order (Lemma 3.1.6, Equation (3.14)), and have the following property:

$$\Re(\lambda_1) \leq \Re(\lambda_2) \leq \cdots \leq \Re(\lambda_{n_\uparrow}) < 0 < \Re(\lambda_{n_\uparrow+2}) \leq \cdots \leq \Re(\lambda_{n_\uparrow+n_\downarrow}).$$

In Kulkarni and Tzenova [75, Theorem 4.2] the solution for (3.24) is given by:

$$f(x) = \sum_{j=1+n_\uparrow}^{n_\downarrow+n_\uparrow} a_j \phi_j^r e^{-\lambda_j x} - \frac{\overline{e}x}{d} + g. \tag{3.28}$$

In this expression g a solution of

$$Tg = -(cR + I)\overline{e}. \tag{3.29}$$

Note that rank $(T) = n - 1$ therefore we have one free variable in g and fix $g_n = 0$ in order to get a solution to (3.29). Coefficients a_j are obtained from the solution to:

$$\sum_{j=1+n_\uparrow}^{n_\downarrow+n_\uparrow} a_j \phi_{ij}^r + g_i = 0, \qquad \forall i \in \mathcal{S}_\downarrow,\ r_i < 0, \tag{3.30}$$

where ϕ_{ij}^r is the ith entry of eigenvector ϕ_j^r.

In Section 2.3.2 we directly use [74, Example 1]. We now extend this for the case where $n_\downarrow \geq 1, \quad n_\uparrow \geq 1$ and $n_0 \geq 0$:

Resulting from the Equation (3.27) we obtain eigenvectors that are partitioned into:

$$\phi^r = \begin{pmatrix} \phi^r_\downarrow \\ \phi^r_0 \\ \phi^r_\uparrow \end{pmatrix}.$$

For the sake of readability we omit the index j in his expression. There are n_0 states with $r_i = 0$ therefore we write:

$$\lambda \begin{pmatrix} R_\downarrow & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & R_\uparrow \end{pmatrix} \begin{pmatrix} \phi^r_\downarrow \\ \phi^r_0 \\ \phi^r_\uparrow \end{pmatrix} = \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} & T_{\downarrow\uparrow} \\ T_{0\downarrow} & T_{00} & T_{0\uparrow} \\ T_{\uparrow\downarrow} & T_{\uparrow 0} & T_{\uparrow\uparrow} \end{pmatrix} \begin{pmatrix} \phi^r_\downarrow \\ \phi^r_0 \\ \phi^r_\uparrow \end{pmatrix},$$

and obtain:

$$\phi^r_0 = -T_{00}^{-1} T_{0\downarrow} \phi^r_\downarrow - T_{00}^{-1} T_{0\uparrow} \phi^r_\uparrow. \tag{3.31}$$

Plugging in (3.31) gives:

$$\lambda \begin{pmatrix} R_\downarrow & 0 \\ 0 & R_\uparrow \end{pmatrix} \begin{pmatrix} \phi^r_\downarrow \\ \phi^r_\uparrow \end{pmatrix} = \begin{pmatrix} T_{\downarrow\downarrow} - T_{\downarrow 0} T_{00}^{-1} T_{0\downarrow} & T_{\downarrow\uparrow} - T_{\downarrow 0} T_{00}^{-1} T_{0\uparrow} \\ T_{\uparrow\downarrow} - T_{\uparrow 0} T_{00}^{-1} T_{0\downarrow} & T_{\uparrow\uparrow} - T_{\uparrow 0} T_{00}^{-1} T_{0\uparrow} \end{pmatrix} \begin{pmatrix} \phi^r_\downarrow \\ \phi^r_\uparrow \end{pmatrix}.$$

The resulting eigenvectors will become:

$$\phi^r = \begin{pmatrix} \phi^r_\downarrow \\ -T_{00}^{-1} \left[T_{0\downarrow} \phi^r_\downarrow + T_{0\uparrow} \phi^r_\uparrow \right] \\ \phi^r_\uparrow \end{pmatrix}. \tag{3.32}$$

Observe that this is equivalent using the eigenvalues and vectors from matrix Q (see Equations 3.2-3.4) and plugging this into (3.32).

In order to have a valid solution only positive eigenvalues can contribute to (3.28). Let Φ be the matrix consisting of all right-eigenvectors ordered according to all corresponding eigenvalues with non negative real parts $\Re(\lambda_{n_\uparrow+1}) = 0 \leq \cdots \leq \Re(\lambda_{n_\downarrow+n_\uparrow})$. We now partition matrix Φ into

$$\Phi = \begin{pmatrix} \Phi_\downarrow \\ \Phi_0 \\ \Phi_\uparrow, \end{pmatrix}, \tag{3.33}$$

where $\Phi_\downarrow$ is $n_\downarrow \times n_\downarrow$, Φ_0 is $n_0 \times n_\downarrow$ and $\Phi_\uparrow$ is $n_\uparrow \times n_\downarrow$.

3.1.10. Definition. We define the conditional expected duration of a busy period and idle period by:

$$\mathbb{E}[C_B] := \Big(\mathbb{E}[\tau_{\mathcal{S}_\downarrow} \mid X(0) = 0, \varphi(0) = i],\ i \in \mathcal{S}_\uparrow\Big), \tag{3.34}$$
$$\tau_{\mathcal{S}_\downarrow} := \inf\{t > 0 : X(t) = 0, \varphi(t) \in \mathcal{S}_\downarrow\},$$
$$\mathbb{E}[C_I] := \Big(\mathbb{E}[\tau_{\mathcal{S}_\uparrow} \mid X(0) = 0, \varphi(0) = i],\ i \in \mathcal{S}_\downarrow\Big), \tag{3.35}$$
$$\tau_{\mathcal{S}_\uparrow} := \inf\{t > 0 : \varphi(t) \in \mathcal{S}_\uparrow\}.$$

3.1.11. Lemma. *The mean duration of a busy period starting in state $i \in \mathcal{S}_\uparrow$ is given by:*

$$\mathbb{E}[C_B] = \Phi_\uparrow \Phi_\downarrow^{-1} g_\downarrow + g_\uparrow, \tag{3.36}$$

where Φ is the block partitioned matrix with right eigen vectors from (3.33) *corresponding to non negative eigenvalues, g is the solution to:*

$$Tg = -(cR + I)\overline{e},$$

with vector g partitioned in

$$g = \begin{pmatrix} g_\downarrow \\ g_0 \\ g_\uparrow \end{pmatrix}.$$

Proof. The solution for (3.24) is given by:

$$f(x) = \sum_{j=1+n_\uparrow}^{n_\downarrow+n_\uparrow} a_j \Phi_j e^{-\lambda_j x} - \frac{\overline{e}x}{d} + g. \tag{3.37}$$

Coefficients a_j are obtained from the solution to:

$$\sum_{j=1+n_\uparrow}^{n_\downarrow+n_\uparrow} a_j \Phi_{ij} + g_i = 0, \qquad \forall i \in \mathcal{S}_\downarrow,\ r_i < 0, \tag{3.38}$$

where Φ_{ij} is the ith entry of jth eigenvector Φ_j in eigenvector matrix Φ. We are interested in the mean first passage time for a busy period started at $x = 0$. Therefore we take $f(0)$:

$$f(0) = \sum_{j=1+n_\uparrow}^{n_\downarrow+n_\uparrow} a_j \Phi_j + g_\uparrow.$$

Switching to matrix notation gives:

$$f(0) = \Phi_\uparrow a + g, \tag{3.39}$$

where

$$\Phi_\downarrow a + g_\downarrow = 0.$$

Matrix $\Phi_\downarrow$ is invertible, therefore we can write:

$$a = -\Phi_\downarrow^{-1} g_\downarrow. \tag{3.40}$$

Plugging (3.40) into (3.39) gives:

$$f(0) = \Phi_\uparrow \Phi_\downarrow^{-1} g_\downarrow + g_\uparrow. \tag{3.41}$$

■

3.1.12. Definition. We define the expected busy cycle time conditioned on starting in a state $i \in \mathcal{S}_\uparrow$ by:

$$\begin{aligned} \mathbb{E}[C_{BB}] &:= \mathbb{E}[\tau_B \mid \varphi(0) = i, X(0) = 0], && i \in \mathcal{S}_\uparrow, \\ \tau_B &:= \inf\{t > \tau_{\mathcal{S}_\downarrow} : \varphi(t) \in \mathcal{S}_\uparrow\}, \\ \tau_{\mathcal{S}_\downarrow} &:= \inf\{t > 0 : X(t) = 0, \varphi(t) \in \mathcal{S}_\downarrow\}. \end{aligned}$$

3.1.13. Lemma. *The overall mean expected busy cycle length is given by:*

$$\mathbb{E}[C] = \pi_B \Big[\mathbb{E}[C_B] + P_{BI} \mathbb{E}[C_I] \Big],$$

where

$$\mathbb{E}[C_I] = -\begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1} \overline{e}, \tag{3.42}$$

resulting in

$$\mathbb{E}[C] = \pi_B \left[\Phi_{\uparrow\downarrow} \Phi_{\downarrow\downarrow}^{-1} g_{\downarrow} + g_{\uparrow} - \begin{pmatrix} \Psi^{\infty} & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1} \overline{e} \right],$$

with $\Psi^{\infty} = \frac{\widehat{C}_1}{\widehat{A}_1}$ *as defined in* (3.11)*.*

Proof. In Lemma 3.1.11 we obtained an expression for the mean busy period. For the idle period using standard first passage time calculations for a CTMC we obtain:

$$\mathbb{E}[C_I] = - \begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1}.$$

Using $\mathbb{E}[C_B]$ and $\mathbb{E}[C_B]$ the expected cycle time can be obtained:

$$\mathbb{E}[C_{BB}] = \mathbb{E}[C_B] + P_{BI}\mathbb{E}[C_I].$$

When a busy period is initiated for a given initiating state $i \in \mathcal{S}_{\uparrow}$ the expected passage time is given by $\mathbb{E}[C_B]$. Remember that in Definition 3.1.7, Equation (3.15) we defined $P_{BI} = \Psi^{\infty}$. From this the expected idle time after a busy period that has been initiated by state $i \in \mathcal{S}_{\uparrow}$ is obtained:

$$\begin{aligned} &\mathbb{E}[\tau_{\mathcal{S}_{\uparrow}} - \tau_0 \mid X(0) = 0, \varphi(0) = i] = P_{BI}\mathbb{E}[C_I], && i \in \mathcal{S}_{\uparrow}, \\ &\tau_{\mathcal{S}_{\uparrow}} := \inf\{t > \tau_0 : \varphi(t) \in \mathcal{S}_{\uparrow}\}, \\ &\tau_0 := \inf\{t > 0 : X(t) = 0\}. \end{aligned} \tag{3.43}$$

This corresponds to taking the expectation over $\mathbb{E}[C_I]$ with respect to the transition matrix P_{BI}. Combining (3.42) and (3.43) gives the expected cycle time given the busy cycle started in state $i \in \mathcal{S}_{\uparrow}$:

$$\mathbb{E}[C_{BB}] = \mathbb{E}[C_B] + P_{BI}\mathbb{E}[C_I], \qquad i \in \mathcal{S}_{\uparrow}.$$

In (3.18) we defined the distribution π_B of states that initiate a busy period. The mean cycle time becomes:

$$\begin{aligned} \mathbb{E}[C] &= \pi_B \Big[\mathbb{E}[C_B] + P_{BI}\mathbb{E}[C_I] \Big], \\ &= \pi_B \left[\Phi_{\uparrow\downarrow} \Phi_{\downarrow\downarrow}^{-1} g_{\downarrow} + g_{\uparrow} - \begin{pmatrix} \Psi^{\infty} & 0 \end{pmatrix} \begin{pmatrix} T_{\downarrow\downarrow} & T_{\downarrow 0} \\ T_{0\downarrow} & T_{00} \end{pmatrix}^{-1} \overline{e} \right]. \end{aligned}$$

■

3.1.14. Theorem. *Let* $M^*(t) := \sup_{0\le s\le t}\{M(s)\}$. *The limiting distribution of* $M^*(t)$ *is given by*

$$\lim_{t\to\infty}\mathbb{P}\{\kappa M^*(t)-\log(bt)\le x\}=\Lambda^{\frac{1}{\mathbb{E}[C]}}(x), \tag{3.44}$$

where

$$b=\pi_B\frac{\widehat{C}_1\widehat{A}_2-\widehat{A}_1\widehat{C}_2}{\widehat{A}_1^2}\overline{e},$$

$$\mathbb{E}[C]=\pi_B\left[\Phi_{\uparrow\downarrow}\Phi_{\downarrow\downarrow}^{-1}g_{\downarrow}+g_{\uparrow}-\begin{pmatrix}\Psi^{\infty} & 0\end{pmatrix}\begin{pmatrix}T_{\downarrow\downarrow} & T_{\downarrow 0}\\ T_{0\downarrow} & T_{00}\end{pmatrix}^{-1}\overline{e}\right]$$

and

$$\kappa=\lambda_{n_\uparrow}.$$

Proof. The proof is similar to that of Iglehart [65, Theorem 3]. In Lemma 3.1.9 we showed that:

$$\lim_{n\to\infty}\mathbb{P}\{\kappa\max_{1\le k\le n}M_+(k)-\log(bn)\le x\}=\Lambda(x).$$

Define $\{c(t):t\ge 0\}$ as the renewal process associated with the length of busy cycles. Then $M^*(t)$ satisfies:

$$\begin{aligned}&\lim_{t\to\infty}\mathbb{P}\{\max_{0\le k\le c(t)}M_+(k)\le x\}\le M^*(t)\le\\ &\lim_{t\to\infty}\mathbb{P}\{\max_{0\le k\le c(t)+1}M_+(k)\le x\}.\end{aligned} \tag{3.45}$$

From Lemma 3.1.13 we know that:

$$\mathbb{E}[C]=\pi_B\left[\Phi_{\uparrow\downarrow}\Phi_{\downarrow\downarrow}^{-1}g_{\downarrow}+g_{\uparrow}-\begin{pmatrix}\Psi^{\infty} & 0\end{pmatrix}\begin{pmatrix}T_{\downarrow\downarrow} & T_{\downarrow 0}\\ T_{0\downarrow} & T_{00}\end{pmatrix}^{-1}\overline{e}\right].$$

Applying the weak law of large numbers for $c(t)$ we obtain:

$$\frac{c(t)}{t}\to\frac{1}{\mathbb{E}[C]},\qquad t\to\infty. \tag{3.46}$$

Using Berman [13, Theorem 3.2] and Lemma 3.1.9 the limiting distribution becomes:

$$\lim_{t\to\infty} \mathbb{P}\{\kappa M^*(t) - \log(bt) \leq x\} = \Lambda^{\frac{1}{\mathbb{E}[C]}}(x). \tag{3.47}$$

The term $\frac{1}{\mathbb{E}[C]}$ from (3.46) represents the expected number of busy cycles per time unit and corresponds to the c in Berman [13, Theorem 3.2]. ■

From Theorem 3.1.14 the expression for the asymptotic distribution for the maximum of fluid queue

$$\mathbb{P}\{M^*(t) > b_{init}\} < p_{empty} \tag{3.48}$$

can now be used to approximate the tail probabilities:

$$\mathbb{P}\{\kappa M^*(t) - \log(bt) > x\} \approx 1 - \Lambda^{\frac{1}{\mathbb{E}[C]}}(x), \tag{3.49}$$

$$\mathbb{P}\{M^*(t) > b_{init}\} \approx 1 - \Lambda^{\frac{1}{\mathbb{E}[C]}}(\kappa b_{init} - \log(bt)), \tag{3.50}$$

whenever we have a sufficiently large b_{init} such that at least $b_{init} > \frac{\log(bt)}{\kappa}$.

Define p_{empty} as the maximal allowed probability that a buffer, with initial contents b_{init}, will become empty during play-out of a video stream of length $t = T_{play}$. Given p_{empty}, that represents the maximum probability a video is disturbed during T_{play}, the initial buffer size b_{init} should be chosen such that

$$b_{init} > \frac{-\log[-\frac{\mathbb{E}[C]}{bT_{play}}\log(1-p_{empty})]}{\kappa}. \tag{3.51}$$

This holds when we have T_{play} sufficiently large such that

$$T_{play} > -\log(1-p_{empty})\frac{\mathbb{E}[C]}{b}.$$

Furthermore $M^*(T_{play})$ represents the limiting distribution on the maximum congestion over time. Then if we consider $M^*(T_{play})$ it should hold that:

$$\mathbb{P}\{M^*(T_{play}) > b_{init}\} < p_{empty}. \tag{3.52}$$

Using the fact that when $t \to \infty$ the maximum M^* converges to a Gumbel distribution the following asymptotic expectation of the maximum level can be derived:

$$\mathbb{E}[M^*(t)] \to \frac{\log\left(\frac{bt}{\mathbb{E}[C]}\right) + \gamma}{\kappa}, \qquad t \to \infty, \tag{3.53}$$

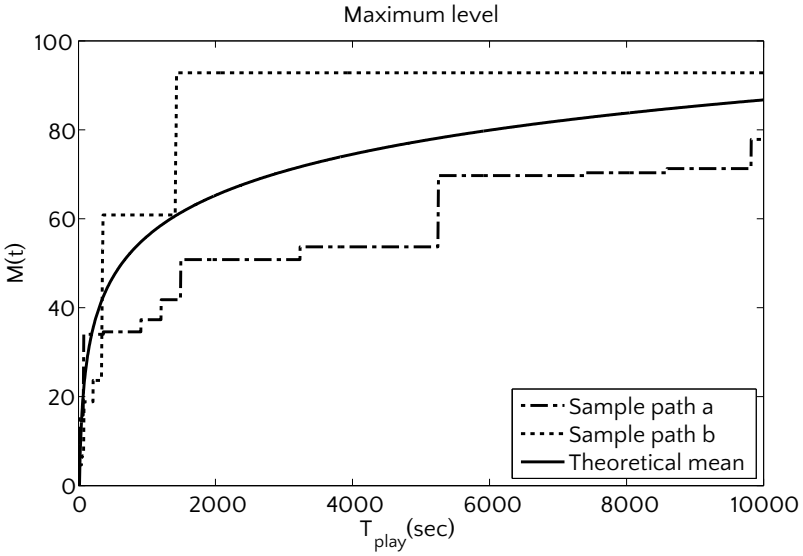
where $\gamma \approx 0.577215665$ is the Euler-Mascheroni constant. Observe that $\mathbb{E}[M^*(t)]$ grows logarithmically over time with logarithmic slope $\frac{1}{\kappa}$.

3.2 Numerical experiments

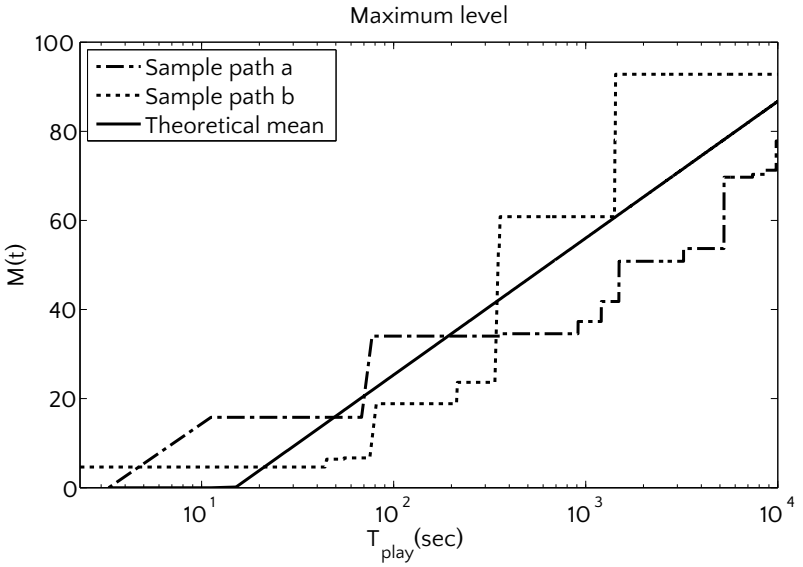
In Section 2.3 we derived that the combined buffer contents, that is congested and in the play-out buffer $X(t) + Y(t) = M^*(t)$, equals the maximum of the congestion process $X(t)$. Moreover the distribution $M^*(t)$ can be approximated by an extreme value distribution for sufficiently large t. From this we derived a mapping from the maximum buffer under-run probability p_{empty} and streaming video duration T_{play} to minimal initial buffer level b_{init}. We will now run simulations in order to evaluate the accuracy of our mapping. Our parameter setting is as follows: $\alpha_1 = 0.1$, $\alpha_2 = 0.2$, $s_1 = 8Mbps$, $s_2 = 2Mbps$, $R_{play} = 4Mbps$, $r_1 = -4$, $r_2 = 2$, $R = \text{diag}\left(\begin{bmatrix} r_1 & r_2 \end{bmatrix}\right)$ and

$$T = \begin{bmatrix} -\alpha_1 & \alpha_1 \\ \alpha_2 & -\alpha_2 \end{bmatrix}.$$

The simulation consists of $1,000,000$ sample paths. Examples of realizations of sample paths are represented in Figure 3.1. In these figures we observe that the sample paths follow the asymptotic mean quite well. Figure 3.1b is the logarithmic time scale variant of Figure 3.1a. On the logarithmic time scale in Figure 3.1b the logarithmic growth behavior of the sample paths with respect to time t can be observed.



(a) *Linear time scale.*



(b) *Logarithmic time scale.*

Figure 3.1: *Sample paths of $M^*(t)$ compared to asymptotic mean as expressed in* (3.53)*. Sample paths a and b correspond to realisations of $X(t) + Y(t)$ from the fluid model simulation.*

In Figure 3.2 simulations ran for different values of T_{play} for fixed R_{play}. On the vertical axis the required buffer (in seconds) is matched with corresponding p_{empty} on the horizontal axis. With Figure 3.3 the required buffer from simulation is compared to the required buffer using our asymptotic result. Here we observe that for reasonably long T_{play} (minutes) the asymptotic result gives a good handle for determining the required buffer time.

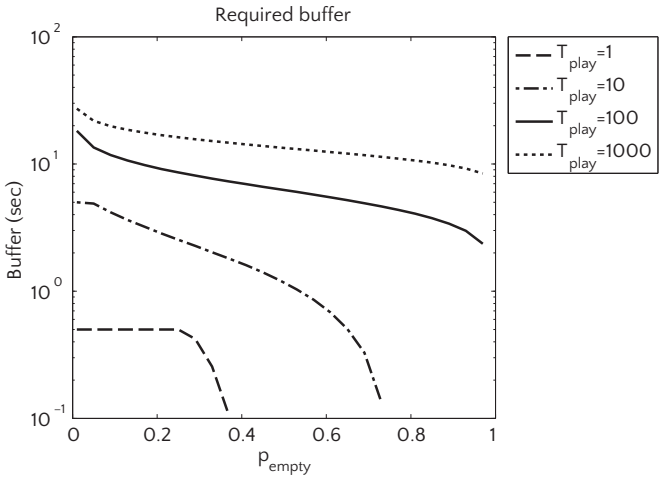


Figure 3.2: *Required buffer (from simulations) for given T_{play} and p_{empty}.*

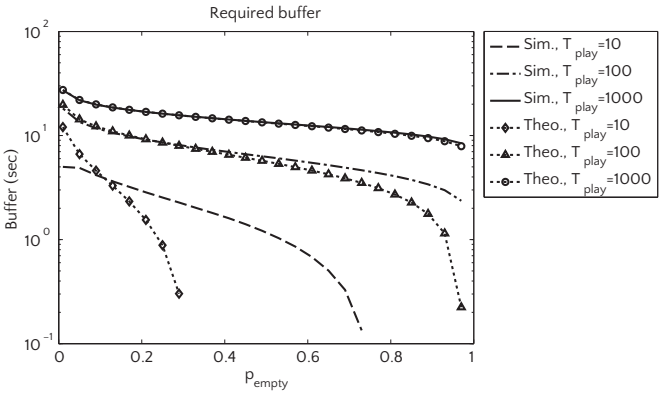
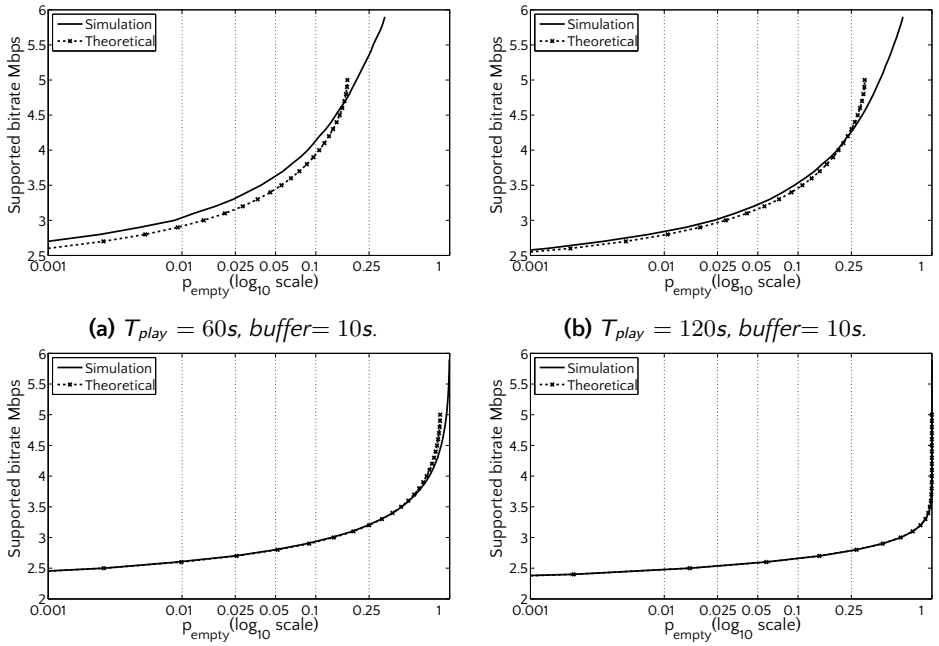


Figure 3.3: *Required buffer (from simulations) for given T_{play} and p_{empty} compared to theoretical required level given by* (3.51)*.*

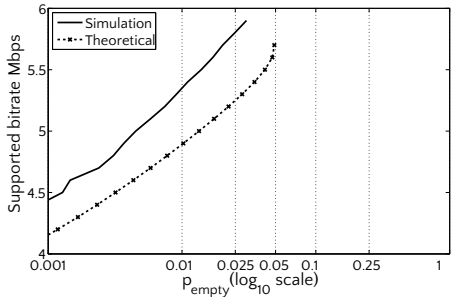
In Figures 3.4a–3.5e the buffer time is fixed while the maximum supported video bitrate is determined. In this setting the network parameters remain fixed while the parameter R_{play} is varied from 2.1 to 5.9. This range is determined by the fact that for the given parameters a minimal bit rate of 2 Mbps is achieved and the average bit rate is equal to 6 Mbps. The maximum supported level determined by simulation is compared to the theoretical maximum supported bit rate. Using (3.50) for given parameters (including R_{play}) the empty buffer probability p_{empty} can be approximated. Note that κ, b and $\mathbb{E}[C]$ all depend on R_{play}. Finding a supported R_{play} using (3.50) is done by applying a search method.



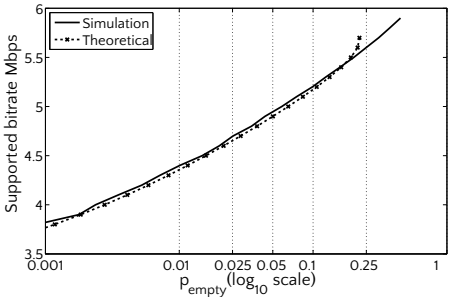
(a) $T_{play} = 60$*s*, *buffer*$= 10$*s*. **(b)** $T_{play} = 120$*s*, *buffer*$= 10$*s*.

(c) $T_{play} = 600$*s*, *buffer*$= 10$*s*. **(d)** $T_{play} = 3600$*s*, *buffer*$= 10$*s*.

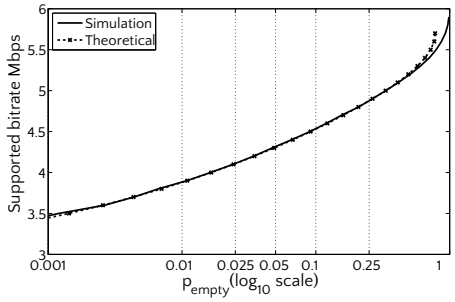
Figure 3.4: *Supported bitrate for given T_{play} and initial buffer level (in seconds) with respect to p_{empty}.*



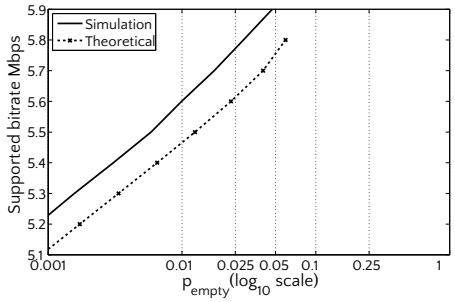
(a) $T_{play} = 120s$, *buffer*$= 30s$.



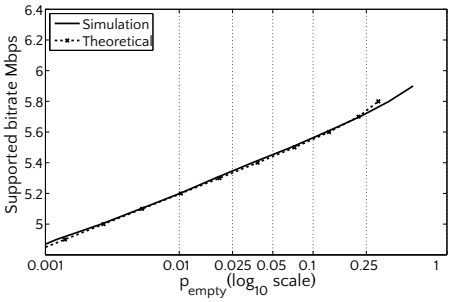
(b) $T_{play} = 600s$, *buffer*$= 30s$.



(c) $T_{play} = 3600s$, *buffer*$= 30s$.



(d) $T_{play} = 600s$, *buffer*$= 60s$.



(e) $T_{play} = 3600s$, *buffer*$= 60s$.

Figure 3.5: *Supported bitrate for given T_{play} and initial buffer level (in seconds) with respect to p_{empty}.*

3.3 Discussion

We extended our model in Chapter 2 such that it supports more than two rates in the Markov Modulated fluid model. In this case there is not always an explicit expression for the initial level of the play-out-buffer. A complicating factor is that the

inverse is needed of a complicated matrix expression from the fluid model equations in Equation 2.11, which resulted in a spectral theory analysis approach. However using the Binet-Cauchy formula on minors [47] (Page 12) we were able to provide an explicit recipe to calculate the asymptotic behavior of the maximum level in the Markov Modulated fluid queue. The result can directly be plugged into Equation 2.39 from Section 2.4 to dimension the initial play-out buffer size. From the simulation results we observe that for reasonably long T_{play} the asymptotic result gives a good handle on the required buffer time. The longer the video stream the more accurately the asymptotic distribution of the maximum corresponds to the real distribution of the maximum.

The speed of convergence to the extreme value distribution depends on the rate in which transitions (of the CTMC that models throughput) occur. In the examples we observe that for small timescale the model is less accurate. An improvement would be adding an approximation for the behavior on shorter time scale. We know that when $t \approx 0$ the distribution quantiles grow linearly with respect to transmission rate and initial distribution. We expect a mix of the small timescale linear behavior model and the long time scale extreme value model to become more accurate.

Appendix 3.A Proof of Lemma 3.1.2

3.A.1. Definition. Let A be a $n \times m$ matrix:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}.$$

Then any order-p minor of A will be denoted as:

$$A \begin{pmatrix} i_1 & i_2 & \cdots & i_p \\ k_1 & k_2 & \cdots & k_p \end{pmatrix} := \det \left[\begin{pmatrix} a_{i_1,k_1} & a_{i_1,k_2} & \cdots & a_{i_1,k_p} \\ a_{i_2,k_1} & a_{i_2,k_2} & \cdots & a_{i_2,k_p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_p,k_p} & a_{i_p,k_2} & \cdots & a_{i_p,k_p} \end{pmatrix} \right],$$

provided that

$$1 \leq i_1 < i_2 < \cdots < i_p \leq m,$$
$$1 \leq k_1 < k_2 < \cdots < k_p \leq n,$$
$$p \leq m, n.$$

The Binet-Cauchy formula on minors [47] (Page 12):
Let A be an $m \times n$ matrix, B be a $n \times q$ matrix and C be an $m \times q$ matrix and $C = AB$. Then any minor of C of order p is the sum of the products of all possible minors of A with order p and corresponding minors of the same order of B:

$$C \begin{pmatrix} i_1 & i_2 & \cdots & i_p \\ j_1 & j_2 & \cdots & j_p \end{pmatrix} = \sum_{1 \le k_1 < k_2 < \cdots < k_m \le n} A \begin{pmatrix} i_1 & i_2 & \cdots & i_p \\ k_1 & k_2 & \cdots & k_p \end{pmatrix} B \begin{pmatrix} k_1 & k_2 & \cdots & k_p \\ j_1 & j_2 & \cdots & j_p \end{pmatrix}.$$

3.A.2. Lemma. *Let A be a $n \times n$ matrix:*

$$A = \sum_{k=1}^{m} A_k,$$

with:

$$A_k = \begin{bmatrix} a_{k,1,1} & \cdots & a_{k,1,n} \\ \vdots & \ddots & \vdots \\ a_{k,n,1} & \cdots & a_{k,n,n} \end{bmatrix}$$

Define $\mathbf{A}$ as a $n \times mn$ matrix with:

$$\mathbf{A} = \begin{bmatrix} A_1 & A_2 & \cdots & A_k \end{bmatrix},$$

and $\mathbf{I}$ is a $mn \times n$ matrix (consisting of m $n \times n$ identity matrices I_n) defined by:

$$\mathbf{I} = \begin{bmatrix} I_n & I_n & \cdots & I_n \end{bmatrix}^T.$$

Let $\mathcal{V}$ be the set of subsets with exactly $n-1$ elements from the set $\{1, 2, \dots, mn\}$ which is defined by:

$$\mathcal{V} = \{(k_1, k_2, \dots, k_{n-1}) : 1 \le k_1 < k_2 < \cdots < k_{n-1} \le mn\}.$$

Then the following holds:

$$\operatorname{adj}(A) = \sum_{v \in \mathcal{V}} \operatorname{adj}\Big(F_C(\mathbf{A}, v) F_R(\mathbf{I}, v)\Big),$$

with operators:

$$F_C(\mathbf{A}, v) = \begin{bmatrix} \mathbf{a}_{1,v_1} & \cdots & \mathbf{a}_{1,v_{n-1}} \\ \vdots & \ddots & \vdots \\ \mathbf{a}_{n,v_1} & \cdots & \mathbf{a}_{n,v_{n-1}} \end{bmatrix},$$

$$F_R(\mathbf{I}, v) = \begin{bmatrix} \mathbf{i}_{v_1,1} & \cdots & \mathbf{i}_{v_1,n} \\ \vdots & \ddots & \vdots \\ \mathbf{i}_{v_{n-1},1} & \cdots & \mathbf{i}_{v_{n-1},n} \end{bmatrix},$$

Operator $F_C(\mathbf{A}, v)$ selects the columns from $\mathbf{A}$ according to vector v, while operator $F_R(\mathbf{I}, v)$ selects rows from $\mathbf{I}$ according to vector v.

Proof. We write $\sum_{k=1}^{m} A_k = \mathbf{AI} = \begin{bmatrix} A_1 & A_2 & \cdots & A_k \end{bmatrix} \begin{bmatrix} I_n & I_n & \cdots & I_n \end{bmatrix}^T$. Using the Binet-Cauchy formula on minors, this can be rewritten to:

$$\begin{aligned} \operatorname{adj}[A] = \operatorname{adj}[\mathbf{AI}] &= \begin{pmatrix} \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{1,1}(v) & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{1,2}(v) & \cdots & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{1,n}(v) \\ \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{2,1}(v) & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{2,2}(v) & \cdots & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{2,n}(v) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{n,1}(v) & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{n,2}(v) & \cdots & \sum_{v\in\mathcal{V}} \bar{\mathbf{a}}_{n,n}(v) \end{pmatrix} \\ &= \sum_{v\in\mathcal{V}} \begin{pmatrix} \bar{\mathbf{a}}_{1,1}(v) & \bar{\mathbf{a}}_{1,2}(v) & \cdots & \bar{\mathbf{a}}_{1,n}(v) \\ \bar{\mathbf{a}}_{2,1}(v) & \bar{\mathbf{a}}_{2,2}(v) & \cdots & \bar{\mathbf{a}}_{2,n}(v) \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\mathbf{a}}_{n,1}(v) & \bar{\mathbf{a}}_{n,2}(v) & \cdots & \bar{\mathbf{a}}_{n,n}(v) \end{pmatrix} \\ &= \sum_{v\in\mathcal{V}} \operatorname{adj}\Big(F_C(\mathbf{A}, v) F_R(\mathbf{I}, v)\Big), \end{aligned}$$

with

$$\bar{\mathbf{a}}_{i,j}(v) = \mathbf{A} \begin{pmatrix} 1 & \cdots & i-1 & i+1 & \cdots & n \\ v_1 & \cdots & v_{i-1} & v_i & \cdots & v_{n-1} \end{pmatrix} \mathbf{I} \begin{pmatrix} v_1 & \cdots & v_{j-1} & v_j & \cdots & v_{n-1} \\ 1 & \cdots & j-1 & j+1 & \cdots & n \end{pmatrix}.$$

■

We are now ready to finalize the proof of Lemma 3.1.2.

Proof. Let $\mathcal{V}$ be the set of subsets with exactly $n-1$ elements from the set $\{1, 2, \dots, mn\}$ which is defined by:

$$\mathcal{V} = \{(k_1, k_2, \dots, k_{n-1}) : 1 \leq k_1 < k_2 < \cdots < k_{n-1} \leq mn\}.$$

We define $\mathcal{P}$ as the set containing all k-permutations of $n-1$ elements from the set $\{1, \dots, n\}$. Furthermore we define $\mathcal{C}$ as the set with all combinations of $n-1$ elements from the set $\{1, \dots, m\}$. For each combination $c \in \mathcal{C}$ we define:

$$\mathbf{A}_c = \begin{bmatrix} A_{c_1} & A_{c_2} & \cdots & A_{c_{n-1}} \end{bmatrix},$$

$$\mathbf{I}_c = \begin{bmatrix} I_n & I_n & \cdots & I_n \end{bmatrix}^T,$$

and thus:

$$\mathbf{A}_c \mathbf{I}_c = \sum_{k \in c} A_k.$$

Next we apply Lemma 3.A.2:

$$\text{adj}[A] = \sum_{v \in \mathcal{V}} \left(\prod_{k \in v} b_{\lceil k/n \rceil} \right) \text{adj} \left(F_C(\mathbf{A}, v) F_R(\mathbf{I}, v) \right),$$

with

$$\mathbf{A} = \begin{bmatrix} A_1 & A_2 & \cdots & A_k \end{bmatrix},$$

and

$$\mathbf{I} = \begin{bmatrix} I_n & I_n & \cdots & I_n \end{bmatrix}^T.$$

Because all matrices A_k have rank 1 the only adjugates that remain are those where there are $n-1$ columns, at $n-1$ different positions, from $n-1$ different A_k matrices. All other combinations of columns result in a matrix with rank $< n-1$ for which the minors of order $n-1$ are zero. Thus the only elements from $\mathcal{V}$ that contribute are those that correspond to any k-permutation of $n-1$ columns from the set $\{1, \dots n\}$ where each column is selected from a distinct matrix A_k, $k = 1, \dots, m$. Note that each selected column remains exactly on its originating column position in the A_k

matrix. As the only combinations consisting of $n-1$ columns at unique positions from $n-1$ unique matrices contribute to non-zero minors it holds that:

$$\begin{aligned}
&\sum_{v\in\mathcal{V}}\left(\prod_{k\in v} b_{\lceil k/n\rceil}\right)\operatorname{adj}\Big(F_C(\mathbf{A},v)F_R(\mathbf{I},v)\Big)\\
&=\sum_{c\in\mathcal{C}}\sum_{p\in\mathcal{P}}\left(\prod_{k\in c} b_k\right)\operatorname{adj}\Big(F_C(\mathbf{A}_c,v_p)F_R(\mathbf{I}_p,v_p)\Big),
\end{aligned}$$

where v_p is the vector that selects the p_ith column from matrix A_{c_i}:

$$(v_p)_i := n(i-1)+p_i, \qquad\qquad i\in\{1,\dots,n-1\},\ p\in\mathcal{P}.$$

We now define $\mathcal{V}_c$ as be the set of subsets with exactly $n-1$ elements from the set $\{1,2,\dots,n(n-1)\}$ which is defined by:

$$\mathcal{V}_c=\{(k_1,k_2,\dots,k_{n-1}) : 1\le k_1<k_2<\cdots<k_{n-1}\le n(n-1)\}.$$

For each combination $c\in\mathcal{C}$ we can do the opposite: add again the terms (corresponding to zero valued minors) from the set $\mathcal{V}_c$ corresponding to columns of $\mathbf{A}_c=\begin{bmatrix}A_{c_1} & \cdots & A_{c_{n-1}}\end{bmatrix}$:

$$\begin{aligned}
&\sum_{c\in\mathcal{C}}\sum_{p\in\mathcal{P}}\left(\prod_{k\in c} b_k\right)\operatorname{adj}\Big(F_C(\mathbf{A}_c,v_p)F_R(\mathbf{I}_p,v_p)\Big),\\
&=\sum_{c\in\mathcal{C}}\left(\prod_{k\in c} b_k\right)\sum_{v\in\mathcal{V}_c}\operatorname{adj}\Big(F_C(\mathbf{A}_c,v)F_R(\mathbf{I}_c,v)\Big),\\
&=\sum_{c\in\mathcal{C}}\left(\prod_{k\in c} b_k\right)\operatorname{adj}\left[\sum_{k\in c} A_k\right].
\end{aligned}$$

■

Efficient Traffic Splitting over Parallel Wireless Networks with Partial Information

4

Up to now, we have considered information flows over a single path through the network. Packet-switched networks offer the possibility to exploit several paths in parallel. Multi-path communication solutions provide a promising means to improve network performance in areas covered by multiple wireless access networks. Today, little is known about how to effectively exploit this potential. In this chapter we study a model where jobs are transferred over multiple parallel networks, each of which is modeled as a processor sharing node. The goal is to minimize the expected transfer time of elastic data traffic by smartly dispatching the jobs to the networks, based on partial information about the numbers of foreground and background jobs in each of the nodes. In the case of full state information, the optimal policy can be derived via standard MDP-techniques, but for models with partial information an optimal solution is hard to obtain. An important requirement is that the splitting algorithm is efficient, yet simple, easy-to-implement, scalable in the number of parallel networks and robust against changes in the parameter settings. We propose a simple index rule for splitting traffic streams based on partial information, and benchmark the results against the optimal solution in the case of full state information. Extensive simulations with real networks show that this method performs extremely well under practical circumstances for a wide range of realistic parameter settings.

4.1 Background

Today, many wireless networks have already closely approached the Shannon limit on channel capacity, leaving complex signal processing techniques room for only modest improvements in the data transmission rate [40]. A powerful alternative to increase the overall data rate then becomes one in which multiple, likely different, networks are used concurrently because (a) the spectrum is regulated among various frequency bands and corresponding communication network standards, and (b) the overall spectrum usage remained to be relatively low over a wide range of frequencies [46]. The concurrent access to multiple networks simultaneously has opened up enormous possibilities for increasing bandwidth, improving reliability, and enhancing Quality of Service (QoS) in areas that are covered by multiple wireless access networks. Currently, the efficient use of multiple networks concurrently is an active area of both research [97] and standardization efforts [54]. However,

[4] This chapter is based on [19] and [14].

despite the enormous potential for quality improvement, only little is known about how to fully exploit this potential. This raises the need for new splitting algorithms for concurrent access that are simple, easy to implement, yet effective.

The effectiveness of splitting data traffic streams is generally believed to increase when detailed information about the state of the system (e.g., the number of flows, measured round-trip times and the network load) is available. In practice, however, there is often no such detailed information available, or at best only some coarse-grained and aggregated statistics. Consequently, a key the challenge is to achieve efficient network utilization levels and good end-user application performance, based on information that is only partially available. At the same time, for the practical usefulness the splitting algorithms are required to be simple, easy-to-implement, scalable in the number of access networks and robust against changes in the parameter settings. We emphasize the importance of this requirement: 'optimal' splitting algorithms based on idealized situations where all detailed information is available – if possible at all – are often too complicated to be practically infeasible.

Processor Sharing (PS) models provide a powerful means to model the bandwidth sharing behavior of elastic traffic streams in TCP-based data networks. A particularly attractive feature of these models is that they abstract from the complex packet-level details of the network, but at the same time maintain the essential factors that determine the data transfer-time performance of elastic data flows. Moreover, the theory of PS models is well-matured and has been successfully applied to model the flow-level behavior of a variety of communication networks, including CDMA 1xEV-DO [18], WLAN [59], UMTS-HSDPA [109] and ADSL [9]. In [59], an analytic flow-level model was presented that explicitly translates the complex and detailed packet-level dynamics of the FTP/TCP/IP-stack over a WLAN into a M/G/1-PS model for the flow-level performance of data transfers.

In the literature, a variety of fundamental and applied studies have been focused to the splitting and scheduling jobs to multiple nodes. The available results and techniques are outlined below. In the context of telecommunication systems, the concurrent use of multiple network resources in parallel was already described for a Public Switched Digital Network (PSDN) [43], where inverse multiplexing was proposed as a technique to perform the aggregation of multiple independent information channels across a network to create a single higher-rate information channel. Various approaches have appeared to exploit multiple transmission paths in parallel. For example, by using multi-element antennas, as adopted by the IEEE 802.11n standard [64], at the physical layer or by switching datagrams at the link layer [31, 72], and also by using multiple TCP sessions in parallel to a file server [95]. In the latter case, each available network transports part of the requested data in a separate TCP session. Previous work has indicated that downloading from multiple networks concurrently may not always be beneficial [52], but in general significant performance improvements can be realized [56, 58, 62]. Under these circumstances of

using a combination of different network types, in particular, the transport layer-approaches, have shown their applicability [62] as they allow appropriate link layer adaptations for each TCP session.

In a queueing-theoretical context, only few papers study partial information models. Bellman [10] was the first to study decision problems with a transition law that is not completely known. He observed that the problem could be transformed into an equivalent full observation problem by augmenting the state space with the set of probability distributions defined on the domain of the unknown quantity (i.e., the unobserved state, or the unknown parameter) and updating it by Bayes' rule. The transformation of the partial information problem to the complete information model, however, comes with added computational difficulties, since policies are defined over a continuum of states. This is the fundamental problem in developing algorithms for computing optimal policies [89]. There is some work in the theoretical domain to characterize the structure of the optimal policy (see, e.g., [25, 2, 103, 79]). Even then, finding the optimal policy computationally for a general Bayesian decision problem is intractable. Approaches dealing with this are to be satisfied with suboptimal solutions or to develop algorithms that can exploit problem characteristics (see, e.g., [78, 90, 116, 57, 24, 30]). We refer to [80, 85, 107, 76] for some surveys on computational techniques.

In this chapter we study a model (depicted in Figure 4.1) consisting of N non-identical parallel networks that are modeled as PS nodes that serve $N+1$ streams of jobs. Node i has processing speed C_i. Stream 0 is called the foreground stream, and streams $1, \ldots, N$ are called the background streams. Jobs of background stream i are served exclusively at PS node i. Each job of the foreground stream has to be routed to exactly one of the PS nodes by the dispatcher. Job sizes are assumed to be exponentially distributed. The goal is to develop a dynamic dispatching policy that minimizes the expected sojourn time of foreground jobs by using information about the numbers of foreground and background jobs at each of the PS nodes. Based on practice, we assume that the dispatcher is not able to distinguish the number of foreground and background jobs in the network, but instead only has information about the *total* number of jobs.

The model under consideration (see Section 4.2 below for details) was also studied in [14], where we addressed this problem through a learning mechanism, where the dispatcher makes a statistical inference on the distribution of the numbers of foreground and background jobs after the each decision. This Bayesian splitting algorithm in [14] was found to be highly effective in dealing with partial information, and its performance was found to be close to the performance of the optimal policy under full state information. However, the Bayesian approach has two main drawbacks: (1) the method is quite complicated and requires in-depth knowledge about stochastic models, which limits its practical usefulness, and (2) the method is not scalable in the number of parallel access networks, N, because it needs the

full-state information MDP solution, which suffers from the curse of dimensionality. This limits the applicability of the Bayesian methods due to memory constraints.

The method presented in this chapter is a simple index rule that is essentially a convex combination of techniques that are found to work well extreme cases: (1) the Weighted Join Shortest Queue (WJSQ) policy that routes foreground flow arrivals to the node where the total number of flows, normalized by the node speed, is minimized, and (2) the Conditional Sojourn Time (CST) approach where the expected sojourn time, conditioned on the total numbers of flows at each of the networks, is minimized. The WJSQ policy is particularly effective when the foreground traffic load tends to saturate the nodes, whereas the CST policy is expected to perform well in systems with low load, or low foreground load situations. The interpolating factor, denoted α $(0 < \alpha < 1)$, represents the ratio of the foreground load and the remaining amount of capacity. When $\alpha \approx 0$ the CST policy is expected to perform well, whereas for $\alpha \approx 1$ the WJSQ policy is expected to perform well. To assess the effectiveness of the CC method, we have performed extensive simulation experiments in a real network simulator, called OPNET [87], that implements the full wireless protocols stack. The results show that the CC method leads to close-to-optimal performance for a wide range of realistic parameter settings.

We emphasize that the main contribution of this chapter lies in (1) its practical usefulness, providing a simple but very effective means to (near-)optimally split elastic traffic streams over wireless networks based on limited information about the state of the system, (2) its scalability with respect to the number of parallel access networks, and (3) the fact the efficiency of the splitting approach is extensively validated by a wide range of real network simulations (rather than simplified queueing simulations) implementing the complex dynamics of full wireless protocol stacks.

The organization of the chapter is as follows. In Section 4.2 we describe the model and introduce the notation. In Section 3 we discuss the full-state information model and present our simple index-rule based heuristic. In Section 4.4 we discuss the results of extensive numerical evaluation of the heuristic in realistic network simulations with OPNET [87], where full wireless protocol stack is implemented.

4.2 Model

We study a model consisting of N non-identical parallel networks that are modeled as PS nodes that serve $N + 1$ streams of flows (we refer to [59] for details on the validation and the parameterization of PS models for modeling wireless networks). Stream 0 is called the *foreground* stream, and streams $1, \ldots, N$ are called the *background* streams. From each stream flows arrive according to a Poisson process with arrival rate λ_i, $(i = 0, 1, \ldots, N)$. Flows from background stream i are served exclu-

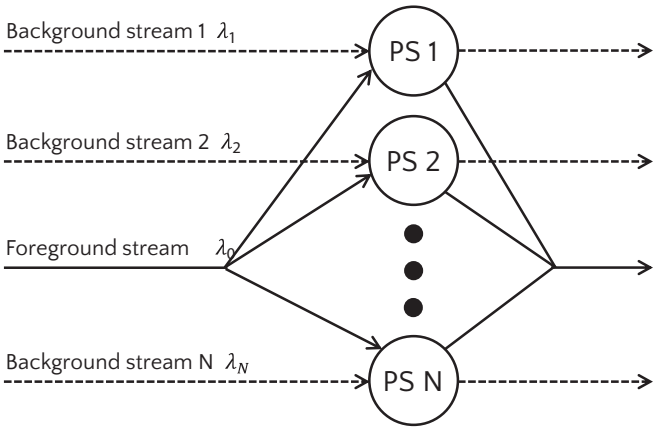


Figure 4.1: *The splitting model.*

sively at PS node i. Each flow from the foreground stream has to be dispatched to one of the PS nodes on the basis of information on the *total* number of flows (thus, number of foreground flows *plus* the number of background flows) at each of the nodes, such that the expected sojourn time $E[S_0]$ for an arbitrary foreground flow is minimized. Flow sizes are assumed to be exponentially distributed with rate μ, and each node has processing speed C_i, so that server i can handle $C_i\mu$ flows per time unit. Without loss of generality, the node capacities are normalized such that $C_1 + \cdots + C_N = 1$. For each node i, the offered background load is given by $\rho_i := \lambda_i / C_i\mu$ $(i = 1, \ldots, N)$, and the foreground load is $\rho_0 := \lambda_0/\mu$. Considering all arriving flows, the total offered load is given by $\rho := \rho_0 + \sum_{i=1}^{N} \rho_i C_i$. For stability reasons, we assume $\rho < 1$. The fraction foreground load compared to the total load is denoted by $\beta := \rho_0/\rho$.

In general, for each given splitting policy that basis its routing decision on the full state information, the model can be described as a CTMC with state space $\mathcal{S} = \mathbb{N}_0^{2N}$, where $\mathbb{N}_0$ is the set of nonnegative integer numbers. Each state $s \in \mathcal{S}$ can be written as $s = (x_1, \ldots, x_N, y_1, \ldots, y_N)$, with x_i the number of foreground flows on the nodes and y_i the number of background flows. In this chapter, it is assumed that the dispatcher has only access to *partial information* in the sense that it has knowledge of $z_i := x_i + y_i$ for $i = 1, \ldots, N$, i.e. the *total* number of flows on each of the nodes. Recall that in the case of full state information the dispatcher has knowledge of $(x_1, \ldots, x_N, y_1, \ldots, y_N)$. Based on the above information, there is a central decision maker that has to decide on the distribution of the foreground jobs over the N servers. In doing so, the aim is to have a decision policy that minimizes $E[S_0]$, where S_0 is the sojourn time of an arbitrary foreground job in the system.

4.3 Splitting algorithms

In this section we describe a number of splitting algorithms, which will be evaluated in the next section. In 4.3.1 we describe the MDP model for the case of full state information (see [94] for details on MDP's), which will be used as a benchmark to assess the efficiency of the index rule for the case of partial information. In Section 4.3.2 we describe both the Bayesian partial information approach 4.3.2.1 and our index rule 4.3.2.2.

4.3.1 Full state information

In this subsection we assume that the dispatcher has full state information, and formulate the optimal dispatching problem as a Markov decision process (MDP). More specifically, the dispatching decisions are based on state description $s = (x_1, \dots, x_N, y_1, \dots, y_N)$ where $s \in \mathcal{S}$ is from the state space $\mathcal{S} = \mathbb{N}_0^{2N}$, and where $\mathbb{N}_0$ is the set of nonnegative integer numbers. Let $\mathcal{I} = \{1, \dots, N\}$ be the set of nodes and $\mathcal{A} = \{1, \dots, N\}$ be the set of actions, where the action i means that the dispatcher forwards the flow to node $i \in \mathcal{A}$. The goal is to minimize the total expected response time by minimizing the total number of active foreground flows. Note that the MDP will not directly obtain the expected sojourn time for foreground flows but by using Little's Law, $\lambda_0 \mathbb{E}[S_0] = \mathbb{E}[N_0]$, we can obtain the average response time from $\mathbb{E}[N_0]$, the average number of foreground flows. The reward function, corresponding to the total number of foreground flows, is defined as $r(s) = x_1 + \cdots + x_N$, where $s = (x_1, \dots, x_N, y_1, \dots, y_N) \in \mathcal{S}$. Furthermore we assume that $\lambda_0 + \cdots + \lambda_N + \mu = 1$ we can always get this by proper scaling. Please note that we already assumed $C_1 + \cdots + C_N = 1$. Let $V(s)$ be the value function, i.e., the asymptotic difference in total costs that results from starting the process in state s 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 (see [94] for details):

$$TV(s) = \min_{i \in \mathcal{A}} \left\{ \lambda_0 V(s + e_i) \right\} \tag{4.1a}$$

$$+ \sum_{i=1}^{N} y_i + \lambda_i V(s + e_{i+N}) \tag{4.1b}$$

$$+ \sum_{i=1}^{N} \mu C_i \frac{x_i}{x_i + y_i} V(s - e_i) \tag{4.1c}$$

$$+ \sum_{i=1}^{N} \mu C_i \frac{y_i}{x_i + y_i} V(s - e_{i+N}). \tag{4.1d}$$

In optimality equation $TV(s)$ (4.1a) corresponds to the foreground flow arrivals that have to be optimized, (4.1b) corresponds to arrivals of cross-traffic flows, (4.1d) corresponds to departures of foreground flows, and (4.1c) corresponds to departures of cross traffic flows.

Applying the backward recursion results in an optimal policy $R^* \in \mathcal{A}^{|\mathbb{S}|}$. An optimal policy contains optimal decisions depending on the number of flows on the PS-nodes. For each state $s = (x_1, \dots, x_N, y_1, \dots, y_N)$, the policy found by the backward recursion $R^*(s) \in \mathcal{A}$ will give an optimal action.

4.3.2 Partial information model

The dynamic server selection model with full information uses a state description $(x_1, \dots, x_N, y_1, \dots, y_N)$ with $2N$ entries. However, in practice, distinguishing the foreground traffic from the background traffic might not be feasible. In these cases, one can only observe the state $(z_1, \dots, z_N)$ with $z_i = x_i + y_i$ for $i = 1, \dots, N$. Now, the dynamic control policy that we derived in the previous section cannot be applied straightforwardly. We now describe two approaches to tackle this problem. First in Section 4.3.2.1 we describe the Bayesian partial information approach. Secondly in Section 4.3.2.2 we describe a near optimal approach based on the conditional sojourn times of the different nodes.

4.3.2.1 Bayesian partial information approach

To apply the control policy one needs to create a mapping from $(z_1, \dots, z_N)$ to $(x_1, \dots, x_N, y_1, \dots, y_N)$, so that (an estimate of the) full information is recovered. Note that it is not sufficient to create a mapping solely based on $(z_1, \dots, z_N)$ at each decision epoch, since it does not use the information contained in the sample path, i.e., many sample paths can lead to the same state $(z_1, \dots, z_N)$. Therefore, we will use Bayesian learning that takes into account the complete history of states in the estimation procedure. We shall call $z = (z_1, \dots, z_N) \in \mathbb{N}_0^N$ the observation state. In order to learn about the division between the number of foreground and background jobs, we will denote by $u_i(n)$ the probability that at server i there are n foreground jobs for $i = 1, \dots, N$. The probability distribution u_i will serve the purpose of information about the states that cannot be observed; hence, $u = (u_1, \dots, u_N)$ is called the belief state. Note that the belief state space is of high dimension, namely $\prod_{i=1}^{N} \{u_i \in [0,1]^{\mathbb{N}_0} \mid \sum_{x \in \mathbb{N}_0} u_i(x) = 1\}$.

Based on the observation and belief states, we construct a state space for the Bayesian dynamic program consisting of the vectors $s = (z, u)$. Note that every arrival and departure gives the system information on how to update the belief state.

Suppose that state s is given and that an arrival of foreground job that is admitted to server i occurs. The new state s_{af_i} is then given by $s_{af_i} = (z + e_i, u')$ where $u_i'(x) = u_i(x-1)$ for $x > 0$ and $u_i'(0) = 0$, and where $u_j'(x) = u_j(x)$ for $j \neq i$. In case of arrival of a background job to server i, we have a new state $s_{ab_i} = (z + e_i, u)$.

In case of departures, we have a similar state transformation. When a foreground job leaves server i, then we have corresponding states $s_{df_i} = ([z - e_i]^+, u')$ with $u_i'(x) = u_i(x+1)$ for $x \geq 0$. Similarly, when a background job leaves server i, then we have $s_{db_i} = ([z - e_i]^+, u)$. Naturally, these transitions cannot be observed, so we take the expectation with respect to the probability distribution u to average over all sample paths. This gives a new dynamic programming operator in which learning is incorporated. This is given by

$$\begin{aligned} TV(s) = \sum_{x_1 \in \mathbb{N}_0} \cdots \sum_{x_N \in \mathbb{N}_0} u_1(x_1) \cdots u_N(x_N) \Big[\sum_{i=1}^N x_i + \sum_{i=1}^N \lambda_i V(s_{ab_i}) \\ + \lambda_0 \min\{V(s_{af_1}), \dots, V(s_{af_N})\} \\ + \sum_{i=1}^N \tfrac{x_i}{z_i} \mu_0 V(s_{df_i}) + \sum_{i=1}^N \tfrac{z_i - x_i}{z_i} \mu_i V(s_{db_i}) \\ + \Big(1 - \lambda_0 - \sum_{i=1}^N \big[\lambda_i + \tfrac{x_i}{z_i}\mu_0 + \tfrac{z_i - x_i}{z_i}\mu_i\big]\Big) V(s) \Big]. \end{aligned} \tag{4.2}$$

Note that the basic idea to transform Equation (4.1) into Equation (4.2) is to take the conditional expectation with respect to the belief state distribution u. Under this condition, the foreground and background jobs can be distinguished so that the structure of the equation resembles the one of the fully observed problem. However, only the transitions to the new belief state need to be adjusted so that the information that has been learned is taken into account. These transitions are provided above.

We end this section with two remarks.

4.3.1. Remark (Complexity)**.** Note that the dynamic programming operator for the Bayesian model (4.2) resembles the dynamic programming operator of the full observation model (4.1). However, the state space of the Bayesian model is of significantly higher dimension as the state variables for the background traffic are continuous. Hence, solving the optimality equation $g + V = TV$ is notoriously hard, both analytically and numerically. In general, the Bayesian updates result in posterior distributions that cannot be captured by a nice structural form. In our problem, however, the decision maker can distinguish foreground and background upon ar-

rival leading to an arrival process with deterministic state transitions. It is only the departures that carry uncertainty with them. This leads to a state transition function, as described above, which keeps the dimensionality of the state space at reasonably low levels. In this way, the structure of the problem makes the Bayesian model a tractable approach (after discretization of the state space). Also note that for arbitrary nodes i and j, the decision as to whether an incoming foreground job should join node i or j, does not depend on the other nodes. Hence, in the decision making one can compare node 1 and 2, take the best node and compare it to node 3, take the best of that comparison and compare it to node 4, and so forth. This leads to a sequence of $N-1$ comparisons. Therefore, the Bayesian approach scales linearly in running time with the number of nodes N.

4.3.2. Remark (Accuracy)**.** In a general Bayesian setting, the belief state represents a probability distribution that represents the likelihood that the process is in a particular state. The accuracy of this estimate, generally, tends to deteriorate as the process progresses due to accumulated errors. In our problem setting, the accuracy of the estimates tends to improve as jobs leave the system. As more jobs leave the system, the support of the posterior distribution reduces to a smaller set of states, limiting the possibilities for errors. In fact, upon departure of the last job in a particular node, the posterior distribution of that node is independent of the past, since the state is exactly known. Thus, all probability mass is concentrated on having 0 jobs in that node. Hence, an empty node leads to a belief state that corresponds to the true state for that node. This observation increases the accuracy of our algorithm due to stability of the system.

4.3.2.2 Conditional sojourn time approach

In this section we propose a heuristic policy for near-optimal dispatching in the case of partial information, i.e. the dispatcher only has knowledge of the total numbers of (foreground plus background) jobs at each node. The policy is based on the combination of two policies that perform well on complementary sets of parameter combinations (see also the discussion below): (1) the Weighted Join the Shortest Queue (WJSQ) policy, and (2) the Conditional Sojourn Time (CST) policy.

The WJSQ policy routes an arriving foreground flow to the node where the total number of flows (normalized by the node speed) is minimal. Thus, the WJSQ forwards an incoming foreground job to node i^*, such that

$$\gamma_{i^*}^{(WJSQ)} = \min\left\{\gamma_1^{(WJSQ)}, \dots, \gamma_N^{(WJSQ)}\right\}, \tag{4.3}$$

where

$$\gamma_i^{(WJSQ)} := \frac{z_i}{C_i} \ (i = 1, \ldots, N).$$

In other words, the WJSQ routes foreground flows to the node with the smallest z_i/C_i ratio. Ties are broken evenly. The WJSQ may be expected to work particularly well when the total load to the system is large (i.e., $\rho \approx 1$) and the foreground load represents a significant fraction of the total load offered to the system (i.e., $\beta \approx 1$).

The CST approach routes an incoming flow to the node for which the expected sojourn time, conditioned on the fact that there are z_i other flows at node i at that moment, is minimal. Using a well-known result for the conditional expected sojourn time in an M/M/1-PS queue [100], the CST policy forwards an incoming foreground job to node i^*, such that

$$\gamma_{i^*}^{(CST)} = \min\left\{\gamma_1^{(CST)}, \ldots, \gamma_N^{(CST)}\right\}, \tag{4.4}$$

where

$$\gamma_i^{(CST)} := \frac{z_i + 2}{2\mu C_i - \lambda_i} \ (i = 1, \ldots, N).$$

The CST approach may be expected to work well if the foreground load is negligible compared to the total load (i.e., $\beta \approx 0$). If the foreground load is large, then the dynamic decision making will induce a correlation between the number of flows in a PS node and the combined arrival process into that node, which leads to a violation of the Poisson assumption that underlies (4.4).

Both the WJSQ and the CST policies generate a switching curve given the total number of flows z_i on each node. We aim to develop a method that works well for the whole range of foreground and background load values. To this end, we propose a method where both switching curves are combined using a convex combination of these curves. The convex combination (CC) approach forwards an incoming foreground job to node i^*, such that

$$\gamma_{i^*}^{(CC)} = \min\left\{\gamma_1^{(CC)}, \ldots, \gamma_N^{(CC)}\right\}, \tag{4.5}$$

where

$$\gamma_i^{(CC)} := \alpha \frac{z_i}{C_i} + (1 - \alpha) \frac{z_i + 2}{2\mu C_i - \lambda_i},$$

and where α $(0 \leq \alpha \leq 1)$ is given by:

$$\alpha := \frac{\rho_0}{\sum_{i=1}^{N} C_i (1 - \rho_i)} \ (i = 1, \ldots, N). \tag{4.6}$$

Thus, the CST method is expected to work well when $\alpha \approx 0$, whereas the WJSQ method is expected to work well when $\alpha \approx 1$. In the next section the CC-approach defined in (4.5)-(4.6) ,and the performance of each of the policies discussed above is evaluated by simulations.

4.4 Numerical experiments

To assess the performance of the index rules discussed in Section 4.3 for efficiently assigning downloads with concurrent access based on partial state information, we have performed extensive experimentation with a state-of-the-art network simulation package OPNET [87], using an implementation for FTP file transfers via TCP/IP over two parallel WLANs. We have performed a large number of experiments with a wide range of parameter settings. The results are outlined below.

4.4.1 Experimental configuration

In the experimental setup all wireless terminals download files from an application server, which may also be a dispatcher in front of several application servers (not shown). The application server has information about the number of ongoing downloads over each of the WLAN access networks, AP1 and AP2, but is unable to distinguish between the multi-homed and the single homed terminals, because there is no binding between both network addresses of the multi-homed terminals. Both WLAN access points operate on non-overlapping frequency channels to establish two non-interfering parallel paths to the application server from the multi-homed systems. The transmission links from the access points towards the application server are considered to incur a negligible delay and loss to packets from and to the access points. This assumption is motivated by the much higher capacities and reliability offered in contemporary fixed-line carrier-grade Internet connections in comparison to the IEEE 802.11b access networks. The analytic model from [59] captures the combined dynamics and protocol overhead of the 802.11 MAC, IP, TCP and application-layer into an explicit expression for the effective service time of a file download. Based on the effective service time, the effective load can be determined of the file transfers in our simulated WLAN networks with a flow-level $M/G/1$ PS model.

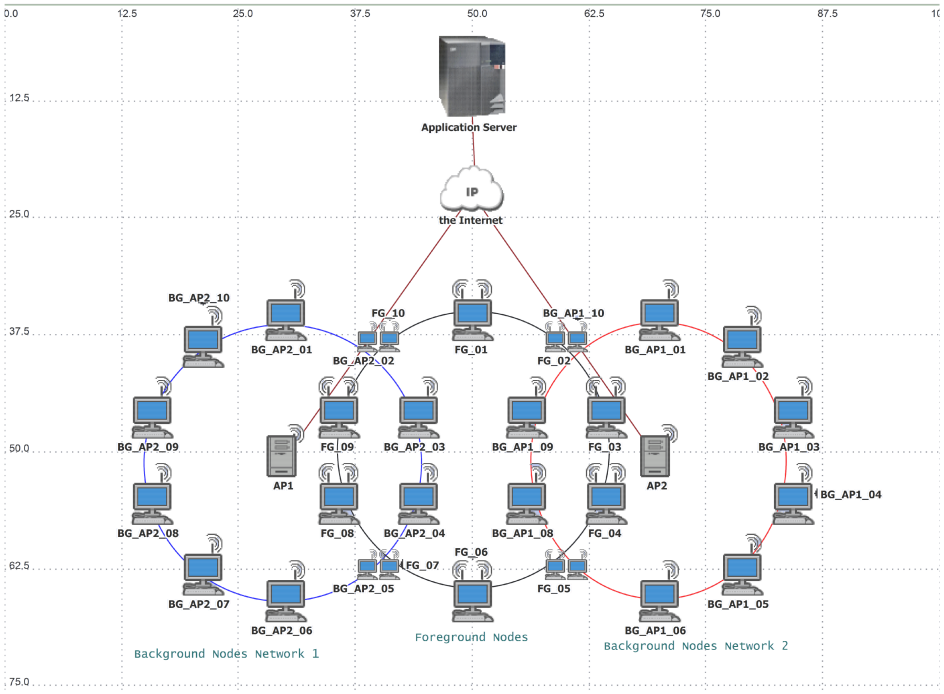


Figure 4.2: *The experimental setup.*

In the simulated network there are ten multi-homed terminals (named $FG\_01 - FG\_10$) that generate download requests (that are considered foreground jobs in the queuing model) with arrival rate λ_0. These foreground terminals are positioned between both access points in a circle with a radius of 15 meter. In addition there are ten single-homed terminals (named with prefix $BG\_AP1\_$) that generate background traffic in network 1 with file downloads arriving with rate λ_1 to the first network. The remaining ten single-homed terminals (named with prefix $BG\_AP2\_$) generate background traffic at rate λ_2 in network 2 in a similar fashion. All background terminals are positioned at an equal distance of 15 meter from their respective access point. The file download requests arrive according to an independent Poisson process and may have multiple file transfers in progress.

The MAC/PHY parameters of the WLAN stations are set in accordance to the widely deployed IEEE 802.11b standard amendment as it relies on the same MAC protocol basis as the contemporary higher rate (IEEE 802.11 a/g/n) amendments and has lower computational requirements for high-load network simulations. Table 4.1 summarizes the IEEE 802.11 MAC parameters used in our analytic model to calculate the effective load values for the simulation runs. In this table, mac is the number

Parameter	Value
mac	224 bits
difs	50 μs
sifs	10 μs
eifs	364 μs
δ	1 μs
R_c	10^6 bps

Parameter	Value
ack	112 bits
R_b	$\{1, 11\} \cdot 10^6$ bps
$Cw_{\min}$	31 slots
phy	192 μs
τ	20 μs

Table 4.1: *IEEE 802.11b MAC parameters.*

of bits of overhead bits associated to a MAC data frame. The difs, sifs, eifs are the DCF, short and extended interframe spacing times, respectively. The δ is the propagation delay that is assumed in our analytic model. R_c is the transmission rate for WLAN acknowledgments of size ack bits, and R_b is the WLAN transmission rate for MAC data frames that is set to 1 or 11 Mbps. $Cw_{\min}$ corresponds to the minimum contention window in slots. Phy is the physical layer overhead, and τ is the slot time. In addition to the WLAN MAC, specific settings apply to the higher protocol layers and are outlined in Table 4.2. In Table 4.2, X_{FTPget} is the size of the FTP GET-

Variable	Setting
X_{FTPget}	4096 bits
$X_{FTPclose}$	64 bits
TCP_{stack}	Full-Featured
X_{MSS}	11584 bits
$X_{tcp/ip}$	416 bits
w	70080 bits (8760 bytes)
X_{file}	$1.6 \cdot 10^6$ bits

Table 4.2: *Network and application settings.*

command that is issued for initiating a file download, $X_{FTPclose}$ is the size of the FTP CLOSE-command that concludes the file transfer at the application. The TCP stack used in our experiments is characterized in OPNET as `Full-Featured', which is an enhanced version of TCP Reno that uses Selective Acknowledgments (SACK) [82] and has a slightly smaller MSS, X_{MSS} (in bits), due to the use of timestamps to fit in the 1500 bytes that are used as the WLAN data frame payload. The number of TCP/IP overhead bits per segment is $X_{tcp/ip}$ bits. The maximum TCP receiver window size is indicated as w (in bits), and the file size as X_{file} (in bits). Based on the parameter setting from Table 4.1 and 4.2 and respecting the engineering guidelines from [59] we can assume that the mean download response times in our simulation model can be accurately predicted from the effective load of the network using the $M/G/1$ PS model.

4.4.2 Experimental results

The OPNET simulations for the experimental results have been run with approximately $322,000$ foreground jobs and the background jobs ranging from roughly $644,000$ jobs to 5.1 million jobs depending on the load. In our simulation study we have considered two scenarios. One simulation scenario considers equal capacity networks in which all terminals are configured to use a WLAN transmission rate of 11 Mbps. For simulating a scenario in which the network capacity of both access network is unequal, the WLAN transmission rate used in AP2 is lowered to 1 Mbps, which reduces the medium capacity for processing file transfers by a factor of 5.79. In this scenario, the background load applied to AP2 is based on the lower capacity, whereas the foreground traffic intensity remains the same as for the equal capacity network. We have executed 48 runs for the equal capacity scenario (24 runs for the fully observed MDP and 24 runs for the heuristics) and 80 runs for the unequal capacity scenario. All runs have completed a total simulation time of 300 hours per run of which 1 hour is the warm-up time leading to a wall clock time of approximately 75 hours per run. This experimental setup is sufficient to derive a 99% confidence interval of approximately 0.7% with respect to the point estimates.

To assess the efficiency of the different partial-information policies, we have simulated the mean transfer time of an arbitrary foreground job, $E[S_0]$, for different policies, and compare the outcome to the full MDP case. For given policy π, the relative error is defined as follows:
For $\pi \in \{\text{WJSQ, CST, CC, Bayes, full MDP}\}$,

$$\Delta\% = \frac{\mathbb{E}\left[S_0|\pi\right] - \mathbb{E}\left[S_0|\text{full MDP}\right]}{\mathbb{E}\left[S_0|\text{full MDP}\right]} \times 100\%. \tag{4.7}$$

Note that the simulations have been run with 10^7 foreground jobs resulting in a 99% confidence interval of approximately 0.1% with respect to the point estimates.

4.4.2.1 The case of equal capacities

We first consider the case where both access networks have the same (normalized) capacity, i.e., $C_1 = C_2$. The results of the experiments are outlined in Tables 4.3 to 4.5, for $\rho_0 = 0.1$ and a number of combinations ρ_1 and ρ_2. Tables 4.3 and 4.4 show the results for the (W)JSQ and the CC policies, benchmarked against the full MDP policy. Table 4.5 shows a comparison between the CC policy and the Bayesian policy [14]. Note that the parameter values are obtained according to the parameterization as defined and validated in [59]. The results in Table 4.3 show that the JSQ policy performs quite well, with a maximum error up to 4.7%. However, the results in Table 4.4 show that the CC policy strongly outperforms JSQ, with a maximum error

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	(0.356, 0.354, 0.6%)	(0.380, 0.369, 3.0%)	(0.401, 0.385, 4.2%)	(0.419, 0.400, 4.7%)	(0.422, 0.406, 4.0%)
	0.2		(0.402, 0.395, 1.8%)	(0.435, 0.420, 3.7%)	(0.463, 0.446, 3.8%)	(0.471, 0.455, 3.5%)
	0.3		(0.422, 0.421, 0.4%)	(0.469, 0.458, 2.3%)	(0.513, 0.498, 2.9%)	(0.533, 0.519, 2.8%)
	0.4			(0.506, 0.501, 1.0%)	(0.574, 0.564, 1.8%)	(0.610, 0.599, 1.9%)
	0.5			(0.548, 0.547, 0.0%)	(0.649, 0.639, 1.6%)	(0.706, 0.696, 1.5%)
	0.6				(0.744, 0.737, 1.0%)	(0.840, 0.828, 1.5%)
	0.7				(0.867, 0.865, 0.2%)	(1.030, 1.018, 1.2%)
	0.8					(1.319, 1.319, 0.0%)

Table 4.3: *Comparison of* $\mathbb{E}[S_0|JSQ]$ *and* $\mathbb{E}[S_0|\textit{full MDP}]$ *for* $\rho_0 = 0.1$.

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	(0.356, 0.354, 0.6%)	(0.371, 0.369, 0.7%)	(0.386, 0.385, 0.2%)	(0.401, 0.400, 0.4%)	(0.408, 0.406, 0.5%)
	0.2		(0.397, 0.395, 0.5%)	(0.422, 0.420, 0.5%)	(0.447, 0.446, 0.1%)	(0.459, 0.455, 0.9%)
	0.3		(0.422, 0.421, 0.4%)	(0.460, 0.458, 0.4%)	(0.501, 0.498, 0.6%)	(0.523, 0.519, 0.8%)
	0.4			(0.503, 0.501, 0.4%)	(0.568, 0.564, 0.8%)	(0.599, 0.599, 0.1%)
	0.5			(0.548, 0.547, 0.0%)	(0.644, 0.639, 0.8%)	(0.702, 0.696, 0.9%)
	0.6				(0.738, 0.737, 0.1%)	(0.835, 0.828, 0.8%)
	0.7				(0.867, 0.865, 0.2%)	(1.022, 1.018, 0.4%)
	0.8					(1.319, 1.319, 0.0%)

Table 4.4: *Comparison of* $\mathbb{E}[S_0|CC]$ *and* $\mathbb{E}[S_0|\textit{full MDP}]$ *for* $\rho_0 = 0.1$.

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	(0.355, 0.356, −0.3%)	(0.370, 0.371, −0.4%)	(0.385, 0.386, −0.2%)	(0.402, 0.401, 0.1%)	(0.408, 0.408, −0.1%)
	0.2		(0.396, 0.397, −0.3%)	(0.420, 0.422, −0.4%)	(0.446, 0.447, 0.0%)	(0.456, 0.459, −0.8%)
	0.3		(0.421, 0.422, −0.3%)	(0.460, 0.460, −0.1%)	(0.502, 0.501, 0.0%)	(0.521, 0.523, −0.3%)
	0.4			(0.503, 0.503, 0.0%)	(0.565, 0.568, −0.6%)	(0.601, 0.599, 0.3%)
	0.5			(0.551, 0.548, 0.6%)	(0.643, 0.644, −0.1%)	(0.700, 0.702, −0.3%)
	0.6				(0.742, 0.738, 0.6%)	(0.831, 0.835, −0.5%)
	0.7				(0.867, 0.867, 0.0%)	(1.028, 1.022, 0.6%)
	0.8					(1.322, 1.319, 0.2%)

Table 4.5: *Comparison of* $\mathbb{E}[S|Bayes]$ *and* $\mathbb{E}[S|CC]$ *for* $\rho_0 = 0.1$.

of 0.9%. The difference in performance between JSQ and CC manifests itself mainly when the background load values are strongly asymmetric. In those case the JSQ policy become highly inaccurate. To illustrate this, consider a two-node system where node 1 has high background load and node 2 has low background load. If $n_1 < n_2$ then the JSQ policy will route an incoming job T to node 1. In this situation, it may well occur that this decision is not optimal, because the sojourn time of T is likely to be stretched due to the background job arrivals at node 1. Table 4.5 shows that the CC policy performs comparably well to the Bayesian policy, despite the fact that the Bayesian policy has a much higher computational complexity. We re-emphasize that the computational complexity of the CC rule is negligible.

4.4.2.2 The case of unequal capacities

Let us now consider the case where the access networks have different capacities, i.e., $C_1 \neq C_2$. To this end, we consider the case $C_1 : C_2 = 1 : 0.17$. The results of the simulations experiments are outlined in Tables 4.6 to 4.8, for $\rho_0 = 0.1$ and a number of combinations ρ_1 and ρ_2. Tables 4.6 and 4.7 show the results for the WJSQ and the CC policies, benchmarked against the full MDP policy (similar to the results in Tables 4.3 and 4.4 for the equal capacity case). Table 4.8 shows a comparison between the CC policy and the Bayesian policy (similar to Table 4.5, see Section 4.2.1). Recall that the parameter values in this setting are obtained according to the parameterization discussed in [59].

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	(1.027, 0.415, 147.5%)	(0.962, 0.415, 131.7%)	(0.865, 0.416, 108.1%)	(0.865, 0.416, 108.1%)	(0.643, 0.415, 54.7%)
	0.2	(1.094, 0.474, 130.7%)	(1.025, 0.475, 115.7%)	(0.930, 0.476, 95.4%)	(0.930, 0.476, 95.4%)	(0.709, 0.475, 49.2%)
	0.3	(1.741, 0.555, 213.8%)	(1.106, 0.555, 99.3%)	(1.013, 0.551, 83.7%)	(1.013, 0.551, 83.7%)	(0.787, 0.555, 41.8%)
	0.4	(1.247, 0.660, 88.9%)	(1.195, 0.664, 80.0%)	(1.111, 0.666, 66.9%)	(1.111, 0.666, 66.9%)	(0.900, 0.664, 35.6%)
	0.5	(1.348, 0.806, 67.1%)	(1.315, 0.818, 60.8%)	(1.252, 0.826, 51.6%)	(1.252, 0.826, 51.6%)	(1.056, 0.833, 26.8%)
	0.6	(1.498, 1.013, 47.8%)	(1.499, 1.047, 43.1%)	(1.463, 1.065, 37.4%)	(1.463, 1.065, 37.4%)	(1.320, 1.099, 20.1%)
	0.7	(1.714, 1.305, 31.3%)	(1.775, 1.390, 27.7%)	(1.828, 1.476, 23.8%)	(1.828, 1.476, 23.8%)	(1.789, 1.595, 12.1%)
	0.8	(2.093, 1.777, 17.8%)	(2.304, 2.013, 14.4%)	(2.545, 2.273, 12.0%)	(2.545, 2.273, 12.0%)	(2.966, 2.862, 3.6%)

Table 4.6: *Comparison of* $\mathbb{E}[S_0|WJSQ]$ *and* $\mathbb{E}[S_0|full\ MDP]$ *for* $\rho_0 = 0.1$ *with* $C_1 = 1$ *and* $C_2 = 0.17$.

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	(0.418, 0.415, 0.8%)	(0.415, 0.415, 0.0%)	(0.416, 0.416, 0.2%)	(0.416, 0.416, 0.2%)	(0.415, 0.415, 0.0%)
	0.2	(0.476, 0.474, 0.3%)	(0.476, 0.475, 0.1%)	(0.477, 0.476, 0.2%)	(0.477, 0.476, 0.2%)	(0.475, 0.475, 0.0%)
	0.3	(0.556, 0.555, 0.3%)	(0.556, 0.555, 0.2%)	(0.555, 0.551, 0.7%)	(0.555, 0.551, 0.7%)	(0.555, 0.555, 0.0%)
	0.4	(0.664, 0.660, 0.6%)	(0.666, 0.664, 0.4%)	(0.667, 0.666, 0.2%)	(0.667, 0.666, 0.2%)	(0.667, 0.664, 0.4%)
	0.5	(0.809, 0.806, 0.3%)	(0.821, 0.818, 0.3%)	(0.831, 0.826, 0.6%)	(0.831, 0.826, 0.6%)	(0.834, 0.833, 0.0%)
	0.6	(1.019, 1.013, 0.6%)	(1.047, 1.047, 0.0%)	(1.070, 1.065, 0.5%)	(1.070, 1.065, 0.5%)	(1.100, 1.099, 0.1%)
	0.7	(1.331, 1.305, 2.0%)	(1.399, 1.390, 0.6%)	(1.477, 1.476, 0.1%)	(1.477, 1.476, 0.1%)	(1.602, 1.595, 0.4%)
	0.8	(1.841, 1.777, 3.6%)	(2.023, 2.013, 0.5%)	(2.313, 2.273, 1.8%)	(2.313, 2.273, 1.8%)	(2.866, 2.862, 0.1%)

Table 4.7: *Comparison of* $\mathbb{E}[S_0|CC]$ *and* $\mathbb{E}[S_0|full\ MDP]$ *for* $\rho_0 = 0.1$ *with* $C_1 = 1$ *and* $C_2 = 0.17$.

The results in Table 4.6 show that the WJSQ policy is highy inefficient when the network capacities are strongly asymmetric, with error even up to over 200%. The results reveal that the WJSQ performs particularly bad in low-load scenarios. However, the results in Table 4.7 show that the CC policy remains to be highly efficient, even when the networks are strongly asymmetric, with a worst-case error less than 4%. Table 4.8 shows again that the CC policy performs comparably well to the Bayesian policy, and that both policies are highly accurate.

		ρ_2				
		0.1	**0.3**	**0.5**	**0.7**	**0.8**
ρ_1	**0.1**	$(0.418, 0.417, 0.4\%)$	$(0.415, 0.415, 0.0\%)$	$(0.416, 0.416, 0.2\%)$	$(0.416, 0.416, 0.2\%)$	$(0.415, 0.415, 0.0\%)$
	0.2	$(0.476, 0.476, -0.1\%)$	$(0.476, 0.477, -0.2\%)$	$(0.477, 0.479, -0.4\%)$	$(0.477, 0.479, -0.4\%)$	$(0.475, 0.475, 0.0\%)$
	0.3	$(0.556, 0.556, 0.1\%)$	$(0.556, 0.556, -0.1\%)$	$(0.555, 0.553, 0.4\%)$	$(0.555, 0.553, 0.4\%)$	$(0.555, 0.555, 0.0\%)$
	0.4	$(0.664, 0.663, 0.2\%)$	$(0.666, 0.665, 0.1\%)$	$(0.667, 0.667, 0.0\%)$	$(0.667, 0.667, 0.0\%)$	$(0.667, 0.667, 0.0\%)$
	0.5	$(0.809, 0.807, 0.3\%)$	$(0.821, 0.819, 0.2\%)$	$(0.831, 0.827, 0.5\%)$	$(0.831, 0.827, 0.5\%)$	$(0.834, 0.835, -0.2\%)$
	0.6	$(1.019, 1.016, 0.3\%)$	$(1.047, 1.052, -0.4\%)$	$(1.070, 1.068, 0.2\%)$	$(1.070, 1.068, 0.2\%)$	$(1.100, 1.100, 0.1\%)$
	0.7	$(1.331, 1.322, 0.7\%)$	$(1.399, 1.410, -0.8\%)$	$(1.477, 1.482, -0.4\%)$	$(1.477, 1.482, -0.4\%)$	$(1.602, 1.597, 0.3\%)$
	0.8	$(1.841, 1.817, 1.3\%)$	$(2.023, 2.057, -1.6\%)$	$(2.313, 2.299, 0.6\%)$	$(2.313, 2.299, 0.6\%)$	$(2.866, 2.876, -0.4\%)$

Table 4.8: *Comparison of* $\mathbb{E}[S_0|Bayes]$ *and* $\mathbb{E}[S_0|CC]$ *for* $\rho_0 = 0.1$ *with* $C_1 = 1$ *and* $C_2 = 0.17$.

4.4.2.3 Varying the asymmetry in foreground versus background load

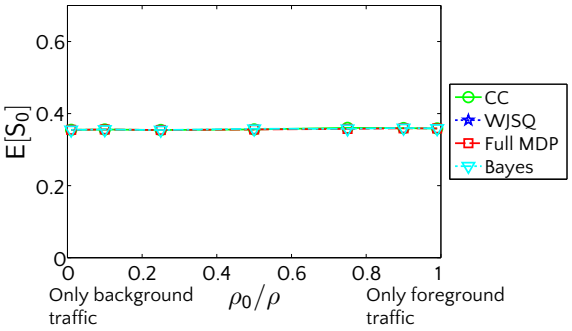
Finally, we check the efficiency of the splitting policies where we vary the fraction of the foreground compared to the total load. Similar to Section 4.2.2, we assume that the ratios of the network capacities are $C_1 : C_2 = 1 : 0.17$. Figures 4.3a to 4.3c show the expected value of the transfer time of an arbitrary foreground job (i.e., $E[S_0]$) as a function of the ratio $\beta = \rho_0/\rho$, where the overall load ρ is kept fixed, for each of the routing policies CC, WJSQ, full MDP and Bayes. Figures 4.3a, 4.3b and 4.3c show the results for $\rho = 0.65$, $\rho = 0.70$ and $\rho = 0.80$, respectively.

The results in Figures 4.3a to 4.3c show again that in all cases the WJSQ policy is strongly outperformed by the other policies. Moreover, we observe that the CC policy, which is based on partial information only, is extremely close to the full MDP solution, which is based on full state information. Also, we observe our simplistic index-based CC rule performs comparably well to the more complicated Bayesian policy. We re-emphasize that the importance of this observation for practical engineering purposes.

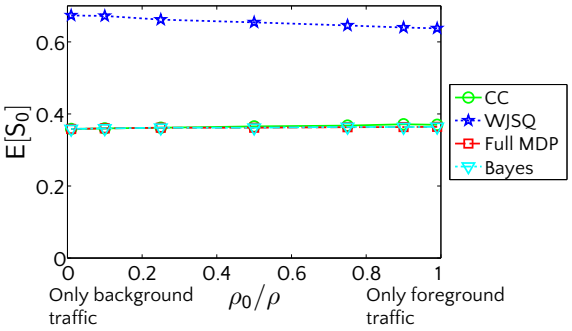
4.5 Discussion

In conclusion, the experimental results demonstrate that the CC-method using partial information strongly outperforms the WJSQ policy, and even leads to close-to-optimal performance that can be obtained using the full-state information MDP. Moreover, the CC method performs equally well when compared to the Bayesian policy, which has a number of drawbacks: (1) it is inherently complicated, which limits its practical usefulness, (2) it is not scalable in the number of access networks N, because it needs the full-state information MDP solution, which suffers from the curse of dimensionality. Typically, this will limit the applicability of the Bayesian methods due to memory constraints. These observations lead to the conclusion

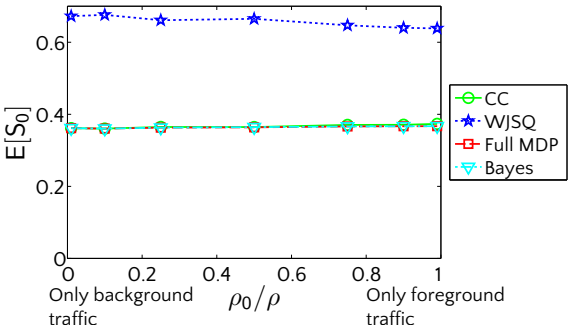
that the CC index rule has a considerable advantage over the Bayesian approach with respect to its practical usefulness and engineering.



(a) $\rho = 0.65$.



(b) $\rho = 0.75$.



(c) $\rho = 0.8$.

Figure 4.3: *Comparison of* $\mathbb{E}[S_0|CC]$, $\mathbb{E}[S_0|\textit{full MDP}]$, $\mathbb{E}[S|WJSQ]$ *and* $\mathbb{E}[S_0|Bayes]$ *in OPNET for different* ρ.

Stochastic Optimal Control for a General Class of Dynamic Resource Allocation Problems

5

Thus far, the considered models and corresponding analyses were focused on optimal usage of available resources. In this chapter we consider a general class of dynamic resource allocation problems within a stochastic optimal control framework. This class of problems arises in a wide variety of applications, each of which intrinsically involves resources of different types and demand with uncertainty and/or variability. The goal is to dynamically allocate capacity for every resource type in order to serve the uncertain/variable demand and maximize the expected net-benefit over a time horizon of interest based on the rewards and costs associated with the different resources. In [48, 49] Xuefeng Gao, Yingdong Lu, Mayank Sharma, and Mark Squillante derived the optimal control policy within a singular control setting, which includes easily implementable algorithms for governing the dynamic adjustments to resource allocation capacities over time.

Based on the singular control policy, we have performed benchmarks to other methods in literature in a realistic setting. Therefore we developed a simulation environment in which experiments are constructed to demonstrate that this control policy is working extremely well. To make the setting more realistic, we analyze Internet-traffic traces and fit these to a demand model that is used in the simulation experiments. Numerical experiments investigate various issues of both theoretical and practical interest, quantifying the significant benefits of our approach over alternative optimization approaches.

5.1 Background

Various canonical forms of general dynamic resource allocation problems arise naturally across a broad spectrum of computer systems and communication networks. As the complexities of these systems and networks continue to grow, together with ubiquitous advances in technology, new approaches and methods are required to effectively and efficiently solve canonical forms of general dynamic resource allocation problems in such complex system and network environments. These environments often consist of different types of resources that are allocated in combination to serve demand whose behavior over time includes different types of uncertainty and variability. Each type of resource has a different reward and cost structure that

[5] This chapter is based on [49].

ranges from the best of a set of primary resource allocation options, having the highest reward, highest cost and highest net-benefit, to a secondary resource allocation option, having the lowest reward, lowest cost and lowest net-benefit. Each type of resource also has different degrees of flexibility and different cost structures with respect to making changes to the allocation capacity. The resource allocation optimization problem we consider consists of adaptively determining the primary and secondary resource capacities that serve the uncertain/variable demand and that maximize the expected net-benefit over a time horizon of interest based on the foregoing structural properties of the different types of resources.

The general class of resource allocation problems studied in this chapter arises in a wide variety of application domains such as cloud computing and data center environments, computer and communication networks, and power-aware (energy-aware) and smart power grid environments, among many others. For example, large-scale cloud computing and data center environments often involve resource allocation over different server options (from fastest performance and most expensive to slowest performance and least expensive) and different network bandwidth options (from guaranteed performance at a cost to opportunistic options at no cost, such as the Internet); e.g., refer to [39, 88, 33, 3, 67]. An additional critical issue in large-scale cloud computing and data center environments concerns the effective and efficient management of the consumption of power by resources in the face of time-varying uncertain system demand and energy prices; e.g., see [55, 83]. Related issues arise in smart power grids where resource allocation is required across a diversity of available energy sources, including a long-term market (stable and less expensive, but rather inflexible), local generation (significant operating constraints and limited capacity), a real-time spot market (readily available and responsive, but at a premium price), and renewables such as wind and solar (less expensive and green, but with high volatility); e.g., refer to [50, 104]. Across these and many other domain-specific resource allocation problems, there is a common need for the dynamic adjustment of allocations among multiple types of resources, each with different structural properties, to satisfy time-varying and uncertain demand.

Motivated by this general class of resource allocation problems, we take a financial mathematics approach that hedges against future risks associated with resource allocation decisions and uncertain demand. Specifically, we consider the underlying fundamental stochastic optimal control problem where the dynamic control policy that allocates primary resource capacity to serve uncertain/variable demand is a variational stochastic process [110] with conditions on its rate of change with respect to time, which in turn determines the secondary resource allocation capacity. The objective is to maximize the expected discounted net-benefit over time based on the structural properties of the different resources types, which we show to be equivalent to a minimization problem involving a piecewise-linear running cost and a proportional cost for making adjustments to the control policy process. Our solution approach is based on first deriving twice continuously differentiable properties

of the value function at the optimal free boundary to determine a solution of the Hamilton-Jacobi-Bellman equation, i.e., the so-called smooth-fit principle. Our theoretical results also include an explicit characterization of the dynamic control policy, which is of threshold type, and then we verify that this control policy is optimal through a martingale argument. In contrast to an optimal static allocation strategy, in which a single primary allocation capacity is determined to maximize expected net-benefit over the entire time horizon, our theoretical results establish that the optimal dynamic control policy adapts its allocation decisions in primary and secondary resources to hedge against the risks of under allocating primary resource capacity (resulting in lost reward opportunities) and over allocating primary resource capacity (resulting in incurred cost penalties).

The research literature covers a great diversity of resource allocation problems, with differing objective functions, control policies, and rewards, costs and flexibility structures. A wide variety of approaches and methods have been developed and applied to address this diversity of resource allocation problems including, for example, online algorithms and dynamic programming. It is therefore important to compare and contrast our problem formulation and solution approach with some prominent and closely related alternatives. One classical instance of a dynamic resource allocation problem is the multi-armed bandit problem [81] where the rewards are associated with tasks and the goal is to determine under uncertainty which tasks the resource should work on, rather than the other way around. Another widely studied problem is the *ski-rental* or *lease-or-buy* problem [44] where there is demand for a resource, but it is initially not known as to how long the resource would be required. In each decision epoch, the choice is between two options: either lease the resource for a fee, or purchase the resource for a price much higher than the leasing fee. Our resource allocation problem differs from this situation in that there are multiple types of resources each with an associated reward and cost per unit time of allocation, since the resources cannot be purchased outright.

From a methodological perspective, the general resource allocation problem we consider in this chapter is closely related to the vast financial mathematics literature on solving stochastic control problems for investment and capacity planning; refer to, e.g., [68, 110, 91]. For example, Beneš et al. [12] consider the so-called *bounded velocity follower problem* with a quadratic running cost objective function, where the authors propose a smooth-fit principle to characterize the optimal policy. In comparison with our study, however, the paper does not consider any costs associated with the actions taken by the control policy, and deals with a smoother objective function. From an applications perspective, there is a growing interest in the computer system and communication network communities to address allocation problems involving various types of resources associated with computation, memory, bandwidth and/or power. For example, Lin et al. [83] consider the problem of dynamically adjusting the number of active servers in a data center as a function of demand to minimize operating costs. In comparison with our study, however,

the paper considers average demand over small intervals of time, subject to system constraints, and develops an online algorithm that is shown to be within a constant factor worse than the corresponding optimal offline policy.

Our study provides important methodological contributions and new theoretical results by deriving the solution of a fundamental singular stochastic optimal control problem. This stochastic optimal control solution approach highlights the importance of timely and adaptive decision making in the allocation of a mixture of different resource options with distinct features in optimal proportions to satisfy time-varying and uncertain demand. Our study also provides important algorithmic contributions through a new class of online policies for dynamic resource allocation problems arising across a wide variety of application domains. Extensive numerical experiments quantify the effectiveness of our optimal online dynamic control algorithm over recent work in the area, including comparisons demonstrating how our optimal online algorithm significantly outperforms the type of optimal offline algorithm within a discrete-time framework recently proposed in [83], which turns out to be related to the optimal online algorithm proposed in [35] within a different discrete-time stochastic optimization framework. This includes relative improvements up to 90% and 130% in comparison with the optimal offline algorithm considered in [83], and significantly larger relative improvements in comparison with the optimal online algorithm in [35].

As a specific application example used for illustrative purposes throughout the chapter, which includes our representative numerical experiments, we shall focus on a basic power-aware resource allocation problem that arises in data center environments. In particular, we consider the problem of dynamically adjusting the allocation of high-performance, high-power servers (primary resources) to serve the uncertain/variable demand within a data center, where any remaining demand is served by low-performance, low-power servers (secondary resources), with the objective of maximizing expected profit (expected rewards minus expected costs). Here, the rewards are based on the performance properties of the type of resources allocated to serve demand over time and the costs are based on the power properties of the type of resources allocated to serve demand over time, together with the costs incurred for making adjustments to the allocation of primary resources over time.

The remainder of this chapter is organized as follows. Section 5.2 defines our mathematical model and formulation of the resource allocation optimization problem. The optimal control policy and corresponding results are presented in Section 5.3, with proofs provided in Section 5.A. A representative sample of numerous numerical experiments are discussed in Section 5.4, followed by some concluding remarks.

5.2 Model

5.2.1 System model

We investigate a general class of resource allocation problems in which different types of resources are allocated to satisfy demand whose behavior over time includes uncertainty and/or variability. To simplify the presentation, we focus on two types of resources: a *primary* resource allocation option that has the highest net-benefit and a *secondary* resource allocation option that has the lowest net-benefit. In terms of our representative application example, the primary resource option consists of high-performance/power servers and the secondary resource option consists of low-performance/power servers. Moreover, high-performance/power server capacity is somewhat less flexible in the sense that its rate of change at any instant of time is bounded, whereas low-performance/power server capacity is more flexible in this regard, each of which is made more precise below. Beyond these differences, both types of resources are capable of serving the demand and all of this demand needs to be served (i.e., no loss of demand). A control policy defines at every time $t \in \mathbb{R}$ the level of primary (high-performance/power) resource allocation, denoted by $P(t)$, and the level of secondary (low-performance/power) resource allocation, denoted by $S(t)$, that are used in combination to satisfy the uncertain/variable demand, denoted by $D(t)$.

Our mathematical resource allocation model generalizes to multiple primary resource allocation options with an analogous net-benefit ordering. Namely, the first primary resource option (highest performance/power servers) has the highest net-benefit, followed by the second primary resource option (next highest performance/power servers) having the next highest net-benefit, and so on, with the (single) secondary resource option (lowest performance/power servers) having the lowest net-benefit. In addition, we have extended our mathematical analysis presented herein to address various forms of this general resource allocation model under certain conditions. However, the single primary and secondary instance of our general resource allocation model captures the key aspects of the fundamental trade-offs among the net-benefits of the various resource allocation options together with their associated risks. We have also shown that the optimal dynamic control policy for various instances of the general model under certain conditions has a very similar structure to that of the single primary and secondary resource model instance. In contrast, the general model, the additional notation, and the technical arguments used to establish these more general results all require much more space than is available to us here. Hence, our focus in this chapter shall be on the canonical single primary (high-performance/power) and secondary (low-performance/power) resource allocation model. The interested reader is referred to [48] for these additional technical details.

We consider the singular stochastic optimal control problem underlying our resource allocation model in which uncertain and/or variable demand needs to be served by primary (high-performance/ power) and secondary (low-performance/power) resource allocation capacities. The demand process $D(t)$ is given by the linear diffusion model

$$dD(t) = bdt + \sigma dW(t),$$

where $b \in \mathbb{R}$ is the demand growth/decline rate (which can be extended to a deterministic function of time, but we do not consider this further in the present chapter), $\sigma > 0$ is the demand volatility/variability, and $W(t)$ is a one-dimensional standard Brownian motion, whose sample paths are nondifferentiable [69, 68]. This demand process is served by the combination of primary (high-performance/power) and secondary (low-performance/power) resource allocation capacities $P(t) + S(t)$. Given the higher net-benefit structure of the primary resource option, the optimal dynamic control policy seeks to determine at every time $t \in \mathbb{R}$ the high-performance/power server allocation capacity $P(t)$ to serve the demand $D(t)$ such that any remaining demand is served by the low-performance/power server allocation capacity $S(t)$.

Let $R_p(t)$ and $C_p(t)$ respectively denote the reward and cost associated with the primary (high-performance/power) resource allocation capacity $P(t)$ at time t. The rewards $R_p(t)$ are linear functions of the primary resource capacity and demand, whereas the costs $C_p(t)$ are linear functions of the primary resource capacity. Therefore, we have

$$R_p(t) = \mathcal{R}_p \times [P(t) \wedge D(t)], \tag{5.1}$$
$$C_p(t) = \mathcal{C}_p \times P(t), \tag{5.2}$$

where $x \wedge y := \min\{x, y\}$, $\mathcal{R}_p \geq 0$ captures all per-unit rewards for serving demand with high-performance/power server capacity, $\mathcal{C}_p \geq 0$ captures all per-unit costs for high-performance/power server capacity, and $\mathcal{R}_p > \mathcal{C}_p$. Observe that the rewards are linear in $P(t)$ as long as $P(t) \leq D(t)$, otherwise any primary resource capacity exceeding demand solely incurs costs without rendering rewards. Hence, from a risk hedging perspective, the risks associated with the primary (high-performance/power) resource allocation position at time t, $P(t)$, concern lost reward opportunities whenever $P(t) < D(t)$ on one hand and concern incurred cost penalties whenever $P(t) > D(t)$ on the other hand.

Since the optimal dynamic control policy serves all remaining demand with secondary (low-performance/power) resource allocation capacity, we therefore have

$$S(t) = [D(t) - P(t)]^+.$$

The corresponding reward function $R_s(t)$ and cost function $C_s(t)$ are then given by

$$R_s(t) = \mathcal{R}_s \times [D(t) - P(t)]^+, \quad (5.3)$$
$$C_s(t) = \mathcal{C}_s \times [D(t) - P(t)]^+, \quad (5.4)$$

where $x^+ := \max\{x, 0\}$, $\mathcal{R}_s \geq 0$ captures all per-unit rewards for serving demand with low-performance/power server capacity, $\mathcal{C}_s \geq 0$ captures all per-unit costs for low-performance/power server capacity, and $\mathcal{R}_s > \mathcal{C}_s$. Hence, from a risk hedging perspective, the secondary (low-performance/power) resource allocation position at time t, $S(t)$, is riskless in the sense that rewards and costs are both linear in the resource capacity actually used.

5.2.2 Problem formulation

The singular stochastic optimal control problem of the previous section allows the dynamic control policy to adapt its allocation positions in primary and secondary resource capacities based on the demand realization observed up to the current time, which we call the risk-hedging position of the dynamic control policy. More formally, the decision process $P(t)$ is adapted to the filtration $\mathcal{F}_t$ generated by $\{D(s) : s \leq t\}$. Furthermore, any adjustments to the primary (high-performance/power) resource allocation capacity have associated costs, where we write $\mathcal{I}_p$ and $\mathcal{D}_p$ to denote the per-unit costs of increasing and decreasing the decision process $P(t)$, respectively; namely, $\mathcal{I}_p$ represents the per-unit cost for increasing the allocation of high-performance/power servers while $\mathcal{D}_p$ represents the per-unit cost for decreasing the allocation of high-performance/power servers. Then the objective of the optimal dynamic control policy is to maximize the expected discounted net-benefit over an infinite horizon, where net-benefit at time t consists of the difference between rewards and costs from primary (high-performance/power) and secondary (low-performance/power) resource allocation capacities minus the additional costs for adjustments to $P(t)$.

In formulating the corresponding stochastic optimization problem, we impose a couple of additional conditions on the variational decision process $\{P(t) : t \geq 0\}$ based on practical aspects of the diverse application domains motivating our study. The control policy cannot instantaneously change the primary (high-performance/power) resource allocation capacity in an attempt to directly follow the demand $D(t)$; i.e., some time is required (even if only a very small amount of time) to adjust $P(t)$. Moreover, the control policy cannot make unbounded adjustments in the primary (high-performance/power) resource allocation capacity at any instant in time; i.e., the amount of change in $P(t)$ at time t is restricted (even if only to a very small extent) by various factors. Given these practical considerations, we assume that the rate of change in the primary resource allocation capacity by the

control policy is bounded. More precisely, there are two finite constants $\theta_\ell < 0$ and $\theta_u > 0$ such that

$$\theta_\ell \;\leq\; \dot{P}(t) \;\leq\; \theta_u,$$

where $\dot{P}(t)$ denotes the derivative of the decision variable $P(t)$ with respect to time.

Now we can present the mathematical formulation of our stochastic optimization problem. Defining

$$\begin{aligned} N_p(t) &:= R_p(t) - C_p(t), \\ N_s(t) &:= R_s(t) - C_s(t), \end{aligned}$$

we seek to determine the optimal dynamic control policy that solves the problem (SC-OPT)

$$\begin{aligned} \max_{\dot{P}(t)} \quad & \mathbb{E}\int_0^\infty e^{-\alpha t}[N_p(t) + N_s(t)]dt \\ & - \mathbb{E}\int_0^\infty e^{-\alpha t}[\mathcal{I}_p \cdot \mathbb{1}_{\{\dot{P}(t) > 0\}}]dP(t) \\ & - \mathbb{E}\int_0^\infty e^{-\alpha t}[\mathcal{D}_p \cdot \mathbb{1}_{\{\dot{P}(t) < 0\}}]d(-P(t)) \qquad (5.5) \\ \text{s.t.} \quad & -\infty \;<\; \theta_\ell \;\leq\; \dot{P}(t) \;\leq\; \theta_u \;<\; \infty, \qquad (5.6) \\ & dD(t) \;=\; b dt + \sigma dW(t), \qquad (5.7) \end{aligned}$$

where α is the discount factor and $\mathbb{1}_{\{A\}}$ denotes the indicator function returning 1 if A is true and 0 otherwise. The control variable is the rate of change in the primary (high-performance/power) resource capacity by the control policy at every time t subject to the lower and upper bound constraints on $\dot{P}(t)$ in (5.6). Note that the second (third) expectation in (5.5) causes a decrease with rate $\mathcal{I}_p$ ($\mathcal{D}_p$) in the value of the objective function whenever the control policy increases (decreases) $P(t)$.

The first expectation in the objective function of the stochastic optimization problem (SC-OPT) can be simplified as follows. Define

$$\begin{aligned} X(t) &:= P(t) - D(t), \\ \mathcal{N}_p &:= \mathcal{R}_p - \mathcal{C}_p, \\ \mathcal{N}_s &:= \mathcal{R}_s - \mathcal{C}_s, \end{aligned}$$

and $x^- := -\min\{x, 0\}$. Upon substituting (5.1), (5.2), (5.3) and (5.4) into the first expectation in (5.5), and making use of the fact that

$$[P(t) \wedge D(t)] \;=\; D(t) - [D(t) - P(t)]^+,$$

we obtain

$$\mathbb{E}\left[\int_0^\infty e^{-\alpha t}[-\mathcal{C}_p X(t) + (\mathcal{N}_s - \mathcal{R}_p)X(t)^-]dt\right] + \mathcal{N}_p\mathbb{E}\left[\int_0^\infty e^{-\alpha t}D(t)dt\right]. \tag{5.8}$$

Since the second expectation in (5.8) does not depend on the control variable $\dot{P}(t)$, this term plays no role in determining the optimal dynamic control policy. Together with the above results, we derive the following stochastic optimization problem which is equivalent to the original optimization problem formulation (SC-OPT):

$$\min_{\dot{P}(t)} \quad \mathbb{E}_x\Bigg[\int_0^\infty e^{-\alpha t}\Big\{\left(\mathcal{C}_+ X(t)^+ + \mathcal{C}_- X(t)^-\right)dt + \left(\mathcal{I}_p\mathbb{1}\left\{\dot{P}(t) > 0\right\} - \mathcal{D}_p\mathbb{1}\left\{\dot{P}(t) < 0\right\}\right)dP(t)\Big\}\Bigg] \tag{5.9}$$

$$\text{s.t.} \quad -\infty < \theta_\ell \leq \dot{P}(t) \leq \theta_u < \infty, \tag{5.10}$$

$$dX(t) = dP(t) - bdt - \sigma dW(t), \tag{5.11}$$

$$X(0) = x, \tag{5.12}$$

$$\mathcal{C}_+ = \mathcal{C}_p, \tag{5.13}$$

$$\mathcal{C}_- = \mathcal{N}_p - \mathcal{N}_s, \tag{5.14}$$

where $\mathbb{E}_x[\cdot]$ denotes expectation with respect to the initial state distribution (i.e., state at time $t = 0$) being x with probability one.

We use $V(x)$ to represent the optimal value of the objective function (5.9); namely, $V(x)$ is the value function of the corresponding stochastic dynamic program. Given its equivalence with the original optimization problem (SC-OPT), the remainder of this chapter will focus on the stochastic dynamic program formulation in (5.9) - (5.14).

5.3 Optimal control policy

In this section we consider our main results on the optimal dynamic control policy for the stochastic optimization problem (5.9) - (5.14). After some technical preliminaries, we present our main results under the conditions $\mathcal{I}_p \geq 0$ and $\mathcal{D}_p \geq 0$, which are likely to be the most interesting case in practice. All other cases of our main results are covered in [48]. Consideration of the proofs of our main results is postponed until the next section.

5.3.1 Preliminaries

To elucidate the exposition, we henceforth assume $b \geq 0$ without loss of generality as one can readily verify that our main results hold when $b < 0$. For notational convenience, we next define the constants

$$r_1 := \frac{b + \sqrt{b^2 + 2\alpha\sigma^2}}{\sigma^2} > 0, \tag{5.15}$$

$$r_2 := \frac{b - \sqrt{b^2 + 2\alpha\sigma^2}}{\sigma^2} < 0, \tag{5.16}$$

$$s_1 := \frac{b - \theta_u + \sqrt{(b-\theta_u)^2 + 2\alpha\sigma^2}}{\sigma^2} > 0, \tag{5.17}$$

$$s_2 := \frac{b - \theta_u - \sqrt{(b-\theta_u)^2 + 2\alpha\sigma^2}}{\sigma^2} < 0, \tag{5.18}$$

$$t_1 := \frac{b - \theta_\ell + \sqrt{(b-\theta_\ell)^2 + 2\alpha\sigma^2}}{\sigma^2} > 0, \tag{5.19}$$

$$t_2 := \frac{b - \theta_\ell - \sqrt{(b-\theta_\ell)^2 + 2\alpha\sigma^2}}{\sigma^2} < 0. \tag{5.20}$$

These quantities are the roots of the quadratic equation

$$\frac{\sigma^2}{2}y^2 + (\theta - b)y - \alpha = 0,$$

when θ takes on the values of θ_ℓ, 0 or θ_u.

Finally, for additional convenience in stating our main results, we further define the following constants

$$\begin{aligned}
B_1 &= (\mathcal{C}_+ - \alpha\mathcal{D}_p)(t_2 - r_2), \\
B_2 &= (\mathcal{C}_- - \alpha\mathcal{I}_p)(s_1 - r_2), \\
B_3 &= (\mathcal{C}_+ + \mathcal{C}_-)(-r_2), \\
A &= (\mathcal{C}_+ + \alpha\mathcal{I}_p)(r_2 - r_1), \\
J_1 &= (\mathcal{C}_+ - \alpha\mathcal{D}_p)(r_1 - t_2), \\
J_2 &= (\mathcal{C}_- - \alpha\mathcal{I}_p)(r_1 - s_1), \\
J_3 &= (\mathcal{C}_+ + \mathcal{C}_-)r_1, \\
K &= (\mathcal{C}_- + \alpha\mathcal{D}_p)(r_2 - r_1),
\end{aligned}$$

in terms of $r_1, r_2, s_1, s_2, t_1, t_2$ given in (5.15) - (5.20). Since $\theta_\ell < 0$ and $\theta_u > 0$, we conclude that B_i and J_i are all positive for $i = 1, 2, 3$, and that A and K are both negative.

5.3.2 Case 1: $\mathcal{D}_p < \mathcal{C}_+/\alpha$ and $\mathcal{I}_p < \mathcal{C}_-/\alpha$

Let us first briefly interpret the conditions of this section. Observe from the objective function (5.9) that $\mathcal{C}_+/\alpha$ reflects the discounted overage cost associated with the primary resource capacity and $\mathcal{C}_-/\alpha$ reflects the corresponding discounted shortage cost, recalling that α is the discount rate. In comparison, $\mathcal{D}_p$ represents the cost incurred for decreasing $P(t)$ when in an overage position while $\mathcal{I}_p$ represents the cost incurred for increasing $P(t)$ when in a shortage position. We now state our main result for this case.

5.3.1. Theorem. *Suppose $\mathcal{D}_p < \mathcal{C}_+/\alpha$ and $\mathcal{I}_p < \mathcal{C}_-/\alpha$. Then there are two threshold values L and U with $L < U$ such that the optimal dynamic control policy is given by*

$$\dot{P}(t) = \begin{cases} \theta_u, & \text{if} \quad P(t) - D(t) < L, \\ 0, & \text{if} \quad P(t) - D(t) \in [L, U], \\ \theta_\ell, & \text{if} \quad P(t) - D(t) > U. \end{cases}$$

Moreover, the values of L and U can be characterized by the following three cases.

I. If

$$0 < \frac{B_3 - B_2}{B_1} < 1$$

and

$$\left(\frac{B_3 - B_2}{B_1}\right)^{\frac{r_2}{r_1}} \geq \frac{J_3 - J_2}{J_1},$$

or

$$B_3 \leq B_2,$$

then we have

$$U > L \geq 0,$$

where L and U are uniquely determined by the two equations:

$$B_1 e^{r_1(L-U)} + J_1 e^{r_2(L-U)} + A = 0, \tag{5.21}$$

$$\frac{B_1 r_2}{r_1 - r_2} e^{r_1(L-U)} + \frac{J_1 r_1}{r_1 - r_2} e^{r_2(L-U)} = (r_1 + r_2 - s_1)(\alpha \mathcal{I}_p + \mathcal{C}_+) + (\mathcal{C}_+ + \mathcal{C}_-) s_1 \cdot e^{s_2 L}. \tag{5.22}$$

II. If

$$\frac{B_3 - B_1}{B_2} > 1$$

and

$$\left(\frac{B_3 - B_1}{B_2}\right)^{\frac{r_2}{r_1}} \geq \frac{J_3 - J_1}{J_2},$$

then we have

$$L < U \leq 0,$$

where L and U are uniquely solved by the two equations:

$$B_2 e^{r_1(U-L)} + J_2 e^{r_2(U-L)} + K = 0, \tag{5.23}$$

$$B_2 \frac{r_2}{r_1 - r_2} e^{r_1(U-L)} + J_2 \frac{r_1}{r_1 - r_2} e^{r_2(U-L)} = (r_1 + r_2 - t_2)(\alpha \mathcal{D}_p + \mathcal{C}_-) + (\mathcal{C}_+ + \mathcal{C}_-) t_2 \cdot e^{t_1 U}. \tag{5.24}$$

III. If none of the above conditions hold, we then have

$$U \geq 0 \geq L,$$

where L and U are uniquely determined by

$$B_1 e^{-r_1 U} + B_2 e^{-r_1 L} = B_3, \tag{5.25}$$

$$J_1 e^{-r_2 U} + J_2 e^{-r_2 L} = J_3. \tag{5.26}$$

Theorem 5.3.1 can be explained as follows. The optimal dynamic control policy seeks to maintain $X(t) = P(t) - D(t)$ within the risk-hedging interval $[L, U]$ at all time t, taking no action (i.e., making no change to $P(t)$) as long as $X(t) \in [L, U]$. Whenever $X(t)$ falls below L, the optimal dynamic control policy pushes toward the risk-hedging interval as fast as possible, namely at rate θ_u, thus increasing the primary (high-performance/power) resource capacity allocation. Similarly, whenever $X(t)$ exceeds U, the optimal dynamic control policy pushes toward the risk-hedging interval as fast as possible, namely at rate θ_ℓ, thus decreasing the primary (high-performance/power) resource capacity allocation. In each of the cases *I*, *II* and *III*, the optimal threshold values L and U are uniquely determined by two nonlinear equations.

5.3.3 Remaining cases

We further establish our main results for all remaining possible conditions on the adjustment costs $\mathcal{D}_p$ and $\mathcal{I}_p$. However, this is not in the scope of this dissertation. Therfore we refer the interested reader to [48] for these additional technical details.

5.4 Numerical experiments

The foregoing sections establish the explicit optimal dynamic control policy among all admissible nonanticipatory control processes $dP(t)$ within a singular stochastic optimal control setting that maximizes the original stochastic dynamic program (SC-OPT) given in (5.5) - (5.7). This optimal dynamic control policy renders a new class of practical online algorithms for general dynamic resource allocation problems that arise in a wide variety of application domains. The resulting online algorithm is easily implementable in computer systems and communication networks (among others) at runtime and consists of maintaining

$$X(t) = P(t) - D(t)$$

within the risk-hedging interval $[L, U]$ at all time t, where L and U are easily obtained in terms of system/network parameters. Extensive numerical experiments have been conducted across a broad spectrum of system/network environments to investigate various issues of both theoretical and practical interest by comparing our online optimal dynamic control algorithm against alternative optimization approaches from recent work in the research literature. In this section, we present a representative sample of these numerical experiments.

The characteristics of the demand process can differ significantly from one system/network environment to the next. However, within a particular environment as well as across a class of similar environments, one often finds consistent seasonal patterns in the average demand process over time, including consistent seasonal effects at daily, weekly, monthly and yearly time scales. We confirm this to be the case for the environments motivating our study through detailed analyses of real-world trace data from a wide variety of proprietary and publicly-available system/network environments. This includes our detailed analysis of request and packet traces from commercial web server environments, university web site proxy servers, e-commerce Internet server environments, and information technology service delivery centers. Based on such detailed analyses of real-world traces, we accurately fitted the average demand process for each environment of interest by a smooth function $f(t)$. Figure 5.1 depicts representative examples of two of these average daily demand patterns $f^1(t)$ and $f^2(t)$, where time t reflects the time zone of

the system/network environment which may be different from that of the demand source. In addition to the average daily demand process, our detailed analyses of real-world traces reveal common seasonal patterns in the volatility of the demand process over time. These demand process volatility patterns tend to be fairly consistent within each system/network environment, whereas they tend to vary much more significantly over an expansive range of values from one environment to the next. Although a great diversity of daily average and volatility demand patterns were discovered throughout our detailed analyses of trace data, the results of numerous numerical experiments comparing our optimal dynamic control policy with alternative optimization approaches under these diverse demand patterns exhibit very similar performance trends, both quantitatively and qualitatively, for a given level of volatility σ. We therefore focus in the remainder of this section on the average daily demand patterns $f^1(t)$ and $f^2(t)$ in Figure 5.1 while varying the volatility parameter σ, noting that the corresponding numerical results are representative of a broad spectrum of system/network environments.

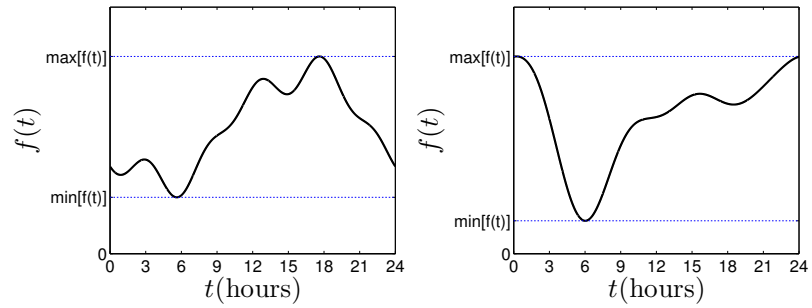


Figure 5.1: *Representative average daily demand patterns $f^1(t)$ (left) and $f^2(t)$ (right).*

To evaluate the benefits of our optimal online dynamic control algorithm in realistic system/network environments, we consider an alternative optimization approach that has recently appeared in the research literature. As previously noted, Lin et al. [83] study a particular optimal offline algorithm for dynamically adjusting the number of active, power-aware servers in a data center to minimize operating costs, and then develop an optimal online algorithm which is proven to be 3-competitive. Our interest for comparison here is in the offline algorithm of [83], which consists of making optimal provisioning decisions in a clairvoyant anticipatory manner based on the known demand within each slot of a discrete-time model where the slot length is chosen to match the timescale at which the data center can adjust its capacity and so that demand activity within a slot is sufficiently nonnegligible in a statistical

sense. Applying this particular optimal offline algorithm within our mathematical framework, we partition the daily time horizon into T slots of length γ such that

$$\begin{aligned} h_i &= (t_{i-1}, t_i], \\ \gamma &= t_i - t_{i-1}, \end{aligned}$$

$i = 1, \ldots, T$, $t_0 := 0$, and we compute the average demand

$$g_i := \gamma^{-1} \int_{h_i} f(t)dt$$

within each slot i yielding the average demand vector

$$(g_1, g_2, \ldots, g_T).$$

Define

$$\Delta(P_i) := P_i - P_{i-1},$$

where P_i denotes the primary (high-performance/power) resource allocation capacity for slot i. The optimal solution under this offline algorithm is then obtained by solving the following linear program (LP):

$$\begin{aligned} \min_{\Delta(P_1),\ldots,\Delta(P_T)} \quad & \sum_{i=1}^{T} \mathcal{C}_+(P_i - g_i)^+ + \mathcal{C}_-(P_i - g_i)^- + \\ & \mathcal{I}_p(P_i - P_{i-1})^+ + \mathcal{D}_p(P_i - P_{i-1})^- \qquad (5.27) \\ \text{s.t.} \quad & -\infty < \theta_\ell \leq \Delta(P_i)/\gamma \leq \theta_u < \infty, \\ & \forall i = 1, \ldots, T, \qquad (5.28) \end{aligned}$$

where the constraints on $\Delta(P_i)$ in (5.28) correspond to (5.10). In this deterministic optimization problem, the control variable is the rate of change in the primary (high-performance/power) resource allocation for each slot i over the daily horizon. We refer to this solution as the offline LP algorithm. This alternative optimization approach is used for comparison in evaluating the benefits of our optimal dynamic control policy.

We also consider a related optimal online algorithm proposed by Ciocan and Farias [35], which is based on a discrete-time model framework similar to that above though within a distinct stochastic optimization framework. We refer to this as the online CF algorithm. Our numerical experiments indicate that the primary (high-performance/power) resource allocation capacity decisions under the online CF algorithm are identical to those under the offline LP algorithm shifted by one time slot.

Hence, the benefits of our optimal online dynamic control algorithm over the online CF algorithm are considerably more significant than those presented here for the offline LP algorithm. We refer the interested reader to [48] for these results and other technical details.

The workloads used for our numerical experiments are generated from real-world trace data taken from various system/network environments. Specifically, once the average daily demand pattern $f(t)$ and the volatility pattern $\sigma(t)$ are extracted from the traces through our detailed analyses, as described above, we construct a linear diffusion process for the entire time horizon such that the drift of the demand process is obtained as the derivative of $f(t)$ (i.e., $b(t) = df(t)$) and the corresponding volatility term is set to match $\sigma(t)$. Since the volatility pattern $\sigma(t)$ tended to be fairly consistent with respect to time within each daily real-world trace for a specific environment and since the volatility pattern tended to vary considerably from one daily real-world trace to another, our linear diffusion demand process is assumed to be governed by the following dynamic model

$$dD(t) = b(t)dt + \sigma dW(t),$$

where we vary the volatility term σ to investigate different system/network environments. The workload for each system/network environment then consists of a set of sample paths generated from the Brownian demand process $D(t)$ defined in this manner. Given such a workload demand process for a specific system/network environment of interest, we calibrate our optimal online dynamic control algorithm by first partitioning the drift function $b(t)$ of the demand process $D(t)$ into piecewise linear segments and then computing the threshold values L and U for each per-segment drift and σ according to Theorem 5.3.1. This (fixed) version of our optimal online dynamic control algorithm is applied to every daily sample path of the Brownian demand process $D(t)$ and the time-average value of net-benefit is computed over this set of daily sample paths. Based on detailed analyses of real-world traces, such applications of our optimal dynamic control policy are easily realized in practice. For comparison under the same set of Brownian demand process sample paths, we compute the average demand vector $(g_1, \ldots, g_T)$ and the corresponding solution under the offline LP algorithm for each daily sample path by solving the linear program (5.27),(5.28) with respect to $(g_1, \ldots, g_T)$, and then we calculate the time-average value of net-benefit over the set of daily sample paths. All of our numerical experiments were implemented in Matlab using, among other functionality, the econometrics toolbox.

We now present a representative sample of our extensive numerical experiments, starting with a first collection of workloads based on the average daily demand pattern $f^1(t)$ illustrated in the top plot of Figure 5.1. Define $f_{\min} := \min_t\{f(t)\}$, $f_{\max} := \max_t\{f(t)\}$, and $f_{\text{avg}} := T^{-1}\int_0^T f(t)dt$. The base parameter settings for this

first set of workloads are: $\alpha = 0.02$, $\sigma = 0.4$, $\theta_l = -10$, $\theta_u = 10$, $\mathcal{C}_+ = 20$, $\mathcal{C}_- = 2$, $\mathcal{D}_p = 0.5$, $\mathcal{I}_p = 0.5$, $f^1_{\min} = 2$, $f^1_{\max} = 7$, $f^1_{\text{avg}} = 4.5$, $x = X(0) = P(0) - D(0) = 0$, and $P_0 = D(0)$. In addition to these base settings, we vary certain parameter values to investigate the impact and sensitivity of these parameters on the performance of both optimization algorithms. This includes conducting numerical experiments under the base parameter settings while varying one of $\sigma \in [0.01, 1.0]$, $\mathcal{C}_+ \in [10, 40]$, $\mathcal{C}_- \in [1, 10]$, $f^1_{\min} \in [1, 5]$, and $f^1_{\max} \in [4, 25]$, all representing parameter values exhibited in real-world system/network environments. For each numerical experiment comprised of a specific workload, we generate $N = 10,000$ daily sample paths using a timescale of a couple of seconds and a γ setting of five minutes, noting that a wide variety of experiments with different timescale and γ settings provided the same performance trends as those presented herein. We then apply our optimal dynamic control policy and the alternative optimization approach to this set of N daily sample paths as described above, where our performance evaluation comparison is based on the expectation of net-benefit realized under each of the two algorithms, also as described above. In particular, the expected net-benefit is computed as the time-average value of the rewards minus the costs from the primary (high-performance/power) and secondary (low-performance/power) resource allocation capacities and minus the costs for adjustments to the primary resource allocation capacity, taken over all N daily sample paths under each of our optimal online dynamic control algorithm and the offline LP algorithm.

Figure 5.2 presents a representative sample of some of our numerical results for the first set of workloads based on $f^1(t)$. The top graph provides performance comparisons of our optimal online dynamic control algorithm against the alternative offline LP algorithm, where the comparisons are based on the relative improvements in expected net-benefit under our optimal control policy as a function of σ; the relative improvement is defined as the difference in expected net-benefit under our optimal dynamic control policy and under the alternative offline LP approach, divided by the expected net-benefit of the latter. For the purpose of comparison across sets of workloads with very different f_{avg} values, we plot this graph as a function of the coefficient of variation CoV $= \sigma / f_{\text{avg}}$. The bottom graph provides similar comparisons of relative improvement in expected net-benefit between our optimal dynamic control policy and the alternative offline LP approach as a function of $\mathcal{C}_+$, both with σ fixed to be 0.4.

We first observe from the top graph in Figure 5.2 that our optimal online dynamic control algorithm outperforms the alternative optimization approach for all $\sigma > 0$. The relative improvements in expected net-benefit under our optimal dynamic control policy grow in an exponential manner with respect to increasing values of σ over the range of CoV values considered, with relative improvements up to 90% in comparison with the offline LP algorithm. The rewards and costs associated with the primary (high-performance/power) and secondary (low-performance/power) resource capacities can be based on either performance or financial metrics, or a

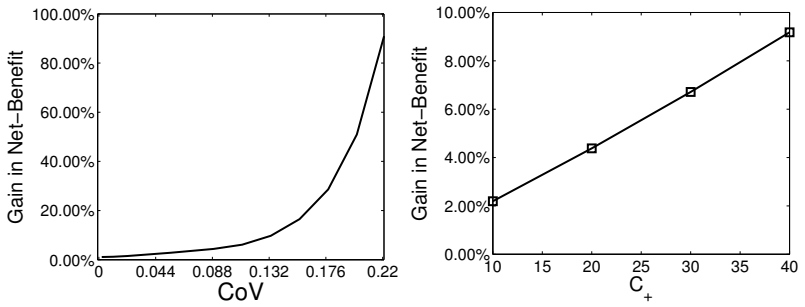


Figure 5.2: *Improvement in expected net-benefit under our optimal dynamic control policy relative to the alternative offline LP algorithm for the first set of workloads based on $f^1(t)$ and for varying values of σ and $\mathcal{C}_+$.*

combination of both. Our results illustrate and quantify the fact that, even in discrete-time models with small time slot lengths γ, nonnegligible volatility plays a critical role in the expected net-benefit of any given resource allocation policy. The significant relative improvements under the optimal online dynamic control algorithm then follow from our stochastic optimal control approach that directly addresses the volatility of the demand process in all primary (high-performance/power) and secondary (low-performance/power) resource allocation decisions. Since our detailed analysis of real-world traces exhibited relatively large CoV values, the results in Figure 5.2 suggest that these net-benefit improvements under our optimal dynamic control policy can be very significant in practice. While the offline LP algorithm based on (5.27),(5.28) would eventually outperform our optimal online dynamic control algorithm as the time slot length γ decreases, we note that the choice for γ in our numerical experiments is considerably smaller than the 10-minute intervals suggested as examples in the literature [83]. Moreover, as discussed in [35], the optimal choice of γ is a complex issue in and of itself and it may need to vary over time depending upon the statistical properties of the demand process $D(t)$. A key advantage of our optimal online dynamic control algorithm is that such parameters are not needed. As explained above, our algorithm can exploit any consistent seasonal patterns for $b(t)$ and $\sigma(t)$ observed from historical traces in order to predetermine the threshold values L and U. In addition, approaches similar to those taken in [35] can be used to adjust these threshold values in real-time based on any nonnegligible changes in the realized values for $b(t)$ and $\sigma(t)$. Furthermore, this latter approach can be used directly for system/network environments whose demand processes do not exhibit consistent seasonal patterns.

We next observe from the bottom graph in Figure 5.2 that the relative improvements in expected net-benefit under our optimal online dynamic control algorithm similarly increases with respect to increasing values of $\mathcal{C}_+$, though in a more lin-

ear fashion. We also note that very similar trends were observed with respect to varying the value of $\mathcal{C}_-$, though the magnitude of the relative improvement in expected net-benefit is smaller. Our numerical experiments suggest that the relative improvements in net-benefit under our optimal dynamic control policy can be more sensitive to $\mathcal{C}_+$ than to $\mathcal{C}_-$. Recall that $\mathcal{C}_+ = \mathcal{C}_p$ is the cost for the primary resource allocation capacity, whereas $\mathcal{C}_- = \mathcal{N}_p - \mathcal{N}_s$ is the difference in net-benefit between the primary (high-performance/power) and secondary (low-performance/power) resource allocation capacities.

We next present a representative sample of our numerical experiments for a second collection of workloads based on the average daily demand pattern $f^2(t)$ illustrated in the bottom plot of Figure 5.1. The base parameter settings for this second set of workloads are: $\alpha = 0.02$, $\sigma = 7.0$, $\theta_l = -100$, $\theta_u = 100$, $\mathcal{C}_+ = 20$, $\mathcal{C}_- = 2$, $\mathcal{D}_p = 0.5$, $\mathcal{I}_p = 0.5$, $f^2_{\text{min}} = 15$, $f^2_{\text{max}} = 90$, $f^2_{\text{avg}} = 61$, $x = X(0) = P(0) - D(0) = 0$ and $P_0 = D(0)$. In addition, analogous to the first set of workloads, we conducted numerical experiments under the base parameter settings while varying one of $\sigma \in [0.01, 15]$, $\mathcal{C}_+ \in [10, 40]$, $\mathcal{C}_- \in [1, 10]$, $f^2_{\text{min}} \in [1, 20]$ and $f^2_{\text{max}} \in [9, 120]$, all representing parameter values exhibited in real-world system/network environments. Once again, for each experiment comprised of a specific workload, we generate $N = 10,000$ sample paths using a timescale of a couple of seconds and a γ setting of five minutes, noting that a wide variety of experiments with different timescale and γ settings provided performance trends that are identical to those presented herein. We then apply our optimal dynamic control policy and the alternative optimization approach to this set of N sample paths as described above. Our performance evaluation comparisons are based on the expectation of net-benefit realized under each of the two algorithms, also as described above.

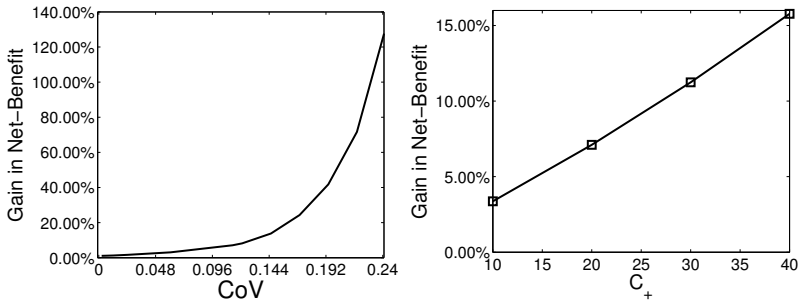


Figure 5.3: *Improvement in expected net-benefit under our optimal dynamic control policy relative to the alternative offline LP algorithm for the second set of workloads based on $f^2(t)$ and for varying values of σ and $\mathcal{C}_+$.*

Figure 5.3 presents a representative sample of some of our numerical results for the second set of workloads based on $f^2(t)$, providing the analogous results that cor-

respond to those in Figure 5.2. We note that the larger range $[f^2_{\text{min}}, f^2_{\text{max}}]$ exhibited in the second average daily demand pattern as well as a higher value of f^2_{avg} lead to both a higher relative net-benefit for fixed σ and a higher sensitivity to changes in σ. Hence, our optimal online dynamic control algorithm has enhanced gains over the alternative offline LP algorithm in terms of expected net-benefit. This relative improvement in expected net-benefit as compared to the set of experiments for the average daily demand pattern $f^1(t)$ can be understood to be caused by the sharp drop in average demand from the maximum value of 90 to a minimum of 15 within a fairly short time span, thus contributing to an increased effective volatility over and above that represented by σ. Therefore, the fact that the relative improvement exhibited by our optimal online dynamic control algorithm is larger under the average daily demand pattern $f^2(t)$, up to 130% in comparison with the offline LP algorithm, can be viewed in a sense to be very much an extension of the finding that the relative improvement provided by our optimal online algorithm increases with an increase in CoV. A similar gain in performance improvement can be seen in the bottom plot when we vary $\mathcal{C}_+$ with a fixed value of σ.

In comparing the results in Figures 5.2 and 5.3 as well as those from our many other numerical experiments, we observe that all of these net-benefit results are quite consistent in terms of qualitative and quantitative trends, where our optimal online dynamic control algorithm consistently provides superior results even when the demand uncertainty/volatility is very small. As an additional observation from our numerical experiments, we find that, upon applying our optimal online dynamic control algorithm and the offline LP algorithm directly to the actual trace data, consistent performance trends among the two optimization approaches were observed in comparison with those shown above for the time-average over many sample paths from the Brownian motion demand process fitted to the same trace data. Finally, though not presented herein, the magnitude of the relative improvements in expected net-benefits exhibited in Figures 5.2 and 5.3 are significantly larger when comparing our optimal online dynamic control algorithm and the online CF algorithm.

5.5 Discussion

In this chapter we investigated a general class of dynamic resource allocation problems arising across a broad spectrum of applications that involve different types of resources and uncertain/variable demand. With a goal of maximizing expected net-benefit based on rewards and costs from the different resources, Xuefeng Gao, Yingdong Lu, Mayank Sharma, and Mark Squillante derived the provably optimal dynamic control policy within a singular stochastic optimal control setting. Their mathematical analysis includes obtaining simple expressions that govern the dy-

namic adjustments to resource allocation capacities over time under the optimal control policy. Based on this analysis, we constructed a wide variety of extensive numerical experiments. The results demonstrate and quantify significant benefits of the optimal dynamic control policy over recently proposed alternative optimization approaches in addressing a general class of resource allocation problems across a diverse range of application domains, including cloud computing and data center environments, computer and communication networks, and power-aware (energy-aware) and smart power grid environments. Moreover, our results strongly suggest that the stochastic optimal control approach taken in this chapter can provide an effective means to develop easily-implementable online algorithms for solving stochastic optimization problems.

Appendix 5.A Proofs

In this section we consider the proofs of the main results. We focus on some aspects of our rigorous proof of Theorem 5.3.1, with all remaining technical details as well as the proofs of the other main results provided in [48]. Our proof proceeds in three main steps. First, we express the optimality conditions for the stochastic dynamic program, i.e., the Bellman equation corresponding to (5.9) - (5.14). We then derive a solution of the Bellman equation and determine the corresponding candidate value function and dynamic control policy. Finally, we verify that this dynamic control policy is indeed optimal through a martingale argument. Each of these main steps is presented in turn.

5.A.1 Proof of Theorem 5.3.1: Step 1

From the Bellman principle of optimality, we deduce that the value function V satisfies for each $t \geq 0$

$$V(x) = \min_{\theta_\ell \leq \dot{P}(t) \leq \theta_u} \mathbb{E}_x \Bigg[\int_0^t e^{-\alpha s} \Big[\Big(\mathcal{C}_+ X(s)^+ + \mathcal{C}_- X(s)^- \Big) ds + (\mathcal{I}_p \mathbb{1}_{\{\dot{P}(s) > 0\}} - \mathcal{D}_p \mathbb{1}_{\{\dot{P}(s) < 0\}}) dP(s) \Big] + e^{-\alpha t} V(X(t)) \Bigg], \tag{5.29}$$

refer to [110, Chapter 4]. Suppose the value function V is smooth, belonging to the set C^2 (i.e., the set of twice continuously differentiable functions) except for a finite number of points, which will be established in step 2. Then, based on a standard

application of Ito's formula as in [73], we derive that the desired Bellman equation for the value function V has the form

$$-\alpha V(x) + \frac{1}{2}\sigma^2 V''(x) - bV'(x) + \mathcal{C}_+ x^+ + \mathcal{C}_- x^- + \inf_{\theta_\ell \le \theta \le \theta_u} \mathcal{L}(\theta, x) = 0, \tag{5.30}$$

where

$$\mathcal{L}(\theta, x) = \begin{cases} (V'(x) + \mathcal{I}_p)\theta & \text{if } \theta \ge 0, \\ (V'(x) - \mathcal{D}_p)\theta & \text{if } \theta < 0. \end{cases} \tag{5.31}$$

5.A.2 Proof of Theorem 5.3.1: Step 2

Our next goal is to construct a convex function Y that satisfies the Bellman equation (5.30) and show that the threshold values L and U are uniquely determined by the corresponding pair of nonlinear equations in Theorem 5.3.1. Suppose a candidate value function $Y(x)$ satisfies (5.30). We then seek to find L and U such that

$$Y'(x) = \begin{cases} \ge \mathcal{D}_p, & \text{if } x \ge U, \\ \in (-\mathcal{I}_p, \mathcal{D}_p) & \text{if } L < x < U, \\ \le -\mathcal{I}_p & \text{if } x \le L. \end{cases} \tag{5.32}$$

Moreover, $Y(x) = O(|x|)$ when $|x|$ goes to ∞, and Y meets smoothly at the points L, 0 and U to order one.

For each of the three cases in Theorem 5.3.1, reflecting different relationships among L, U and 0 based on model parameters, we first establish the desired convexity properties together with the corresponding pair of threshold equations. Case II of Theorem 5.3.1 is considered in the next subsection, noting that Cases I and III and other technical details can be found in [48]. Then, in Section 5.A.2.2, we show that the thresholds L and U can be uniquely determined through these equations.

5.A.2.1 Case II: $0 \ge U > L$

Focusing on the case $0 \ge U > L$ where L and U satisfy (5.23) and (5.24) subject to (5.32), we proceed to solve the Bellman equation (5.30) depending on the value of x in relation to U, 0 and L. There are four subcases to consider as follows.

(i). If $x \ge 0$, we have

$$Y'(x) \ge \mathcal{D}_p$$

and

$$\inf_{\theta_\ell \le \theta \le \theta_u} \mathcal{L}(\theta, x) = \mathcal{L}(\theta_\ell, x).$$

Then the Bellman equation yields

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) + \mathcal{C}_+ x^+ + \mathcal{C}_- x^- + \mathcal{L}(\theta_\ell, x) = 0,$$

or equivalently

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) + \mathcal{C}_+ x + (Y'(x) - \mathcal{D}_p)\theta_\ell = 0.$$

Solving this equation, we derive for $x \ge 0$

$$Y(x) = \frac{\mathcal{C}_+}{\alpha}x + \frac{1}{\alpha}(\frac{\mathcal{C}_+}{\alpha}(\theta_\ell - b) - \mathcal{D}_p\theta_\ell) + l_3 e^{t_2 x}, \tag{5.33}$$

where l_3 is a generic constant to be determined.

(ii). If

$$U < x < 0,$$

we have

$$Y'(x) \ge \mathcal{D}_p,$$

which yields

$$\inf_{\theta_\ell \le \theta \le \theta_u} \mathcal{L}(\theta, x) = \mathcal{L}(\theta_\ell, x),$$

and thus we obtain

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) - \mathcal{C}_- x + (Y'(x) - \mathcal{D}_p)\theta_\ell = 0.$$

This implies for $U < x < 0$

$$Y(x) = -\frac{\mathcal{C}_-}{\alpha}x + \frac{1}{\alpha}(-\frac{\mathcal{C}_-}{\alpha}(\theta_\ell - b) - \mathcal{D}_p\theta_\ell) + \lambda_1 e^{t_1 x} + \lambda_2 e^{t_2 x}, \tag{5.34}$$

where λ_1 and λ_2 are generic constants to be determined.

(iii). If

$$L < x < U,$$

we have

$$-\mathcal{I}_p \leq Y'(x) \leq \mathcal{D}_p$$

and the Bellman equation (5.30) renders

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) - \mathcal{C}_- x = 0.$$

Solving this equation, we deduce for $U > x > L$

$$Y(x) = -\frac{\mathcal{C}_-}{\alpha}x + \frac{b\mathcal{C}_-}{\alpha^2} + \tilde{\lambda}_1 e^{r_1 x} + \tilde{\lambda}_2 e^{r_2 x}, \tag{5.35}$$

where $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$ are generic constants to be determined.

(iv). If $x \leq L$, we have

$$Y'(x) \leq -\mathcal{I}_p$$

and the Bellman equation (5.30) becomes

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) - \mathcal{C}_- x + (Y'(x) + \mathcal{I}_p)\theta_u = 0.$$

The solution is then given by

$$Y(x) = -\frac{\mathcal{C}_-}{\alpha}x + \frac{1}{\alpha}(-\frac{\mathcal{C}_-}{\alpha}(\theta_u - b) + \mathcal{I}_p\theta_u) + l_1 e^{s_1 x}, \tag{5.36}$$

where l_1 is a generic constant to be determined.

Now we match the value and the first-order derivative of Y at the points U, 0 and L. Hence, the function Y that we construct will be twice continuously differentiable

with the exception of at most three points. Let us first consider such matchings at the point U. From (5.34) and (5.35), we derive

$$Y'(U+) = -\frac{\mathcal{C}_-}{\alpha} + \lambda_1 t_1 e^{t_1 U} + \lambda_2 t_2 e^{t_2 U} = \mathcal{D}_p, \tag{5.37}$$

$$Y'(U-) = -\frac{\mathcal{C}_-}{\alpha} + \tilde{\lambda}_1 r_1 e^{r_1 U} + \tilde{\lambda}_2 r_2 e^{r_2 U} = \mathcal{D}_p, \tag{5.38}$$

$$Y(U+) = -\frac{\mathcal{C}_-}{\alpha} U + \frac{1}{\alpha}(-\frac{\mathcal{C}_-}{\alpha}(\theta_\ell - b) - \mathcal{D}_p \theta_\ell) + \lambda_1 e^{t_1 U} + \lambda_2 e^{t_2 U},$$

$$Y(U-) = -\frac{\mathcal{C}_-}{\alpha} U + \frac{b\mathcal{C}_-}{\alpha^2} + \tilde{\lambda}_1 e^{r_1 U} + \tilde{\lambda}_2 e^{r_2 U}.$$

Since

$$Y(U+) = Y(U-),$$

we immediately obtain

$$\tilde{\lambda}_1 e^{r_1 U} + \tilde{\lambda}_2 e^{r_2 U} = \lambda_1 e^{t_1 U} + \lambda_2 e^{t_2 U} - (\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha})\frac{\theta_\ell}{\alpha}. \tag{5.39}$$

Matching at the point L, we deduce [48] from (5.35) and (5.36) expressions for $Y'(L+)$, $Y'(L-)$, $Y(L+)$, $Y(L-)$, and then solve for $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$ in terms of L and the model parameters as follows:

$$\tilde{\lambda}_1 = \frac{\sigma^2}{2\alpha^2} e^{-r_1 L}(1 - \frac{r_1}{r_2})^{-1} B_2, \tag{5.40}$$

$$\tilde{\lambda}_2 = \frac{\sigma^2}{2\alpha^2} e^{-r_2 L}(1 - \frac{r_2}{r_1})^{-1}(-J_2). \tag{5.41}$$

Upon substituting these expressions into (5.38), we obtain (5.23). To establish (5.24), matching Y at the point 0 to order one renders [48] expressions for $Y'(0+)$, $Y'(0-)$, $Y(0+)$, $Y(0-)$, and thus we can derive

$$\lambda_1 = \frac{\sigma^2}{2\alpha^2}(\mathcal{C}_+ + \mathcal{C}_-)\frac{{t_2}^2}{t_1 - t_2}, \tag{5.42}$$

$$\lambda_2 - l_3 = \frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha}(\frac{1}{t_2} + \frac{1}{t_1 - t_2}). \tag{5.43}$$

Upon substituting the expressions for λ_1, $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$ into (5.37) and (5.39), cancelling λ_2, and simplifying the resulting expressions, we can conclude that L and U satisfy (5.24).

Next, we verify that the candidate value function Y satisfies the required first-order properties in (5.32). Since we have constructed the function Y with

$$\begin{aligned} Y'(U) &= \mathcal{D}_p, \\ Y'(L) &= -\mathcal{I}_p, \end{aligned}$$

then to establish (5.32) it suffices to verify the convexity of the function Y. To this end, one first readily confirms that for $x < L$

$$Y''(x) = l_1 s_1^2 e^{s_1 x} = (\frac{\mathcal{C}_-}{\alpha} - \mathcal{I}_p) s_1 e^{s_1(x-L)}.$$

Given that $\mathcal{I}_p < \mathcal{C}_-/\alpha$ and that $s_1 > 0$ in (5.17), we can conclude

$$Y''(x) > 0 \quad \text{for } x < L.$$

For $x \in [L, U]$, we substitute (5.40) and (5.41) for $\tilde{\lambda}_1$ and $\tilde{\lambda}_2$ into (5.35), from which we deduce [48]

$$Y''(x) \;>\; 0.$$

Turning to $x \in (U, 0]$, we derive [48] from (5.37)

$$\begin{aligned} Y''(x) &= \lambda_1 t_1^2 e^{t_1 x} + \lambda_2 t_2^2 e^{t_2 x}, \\ &= \lambda_1 t_1^2 e^{t_1 x} + (\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha} - \lambda_1 t_1 e^{t_1 U}) t_2 e^{t_2(x-U)}, \\ &= (\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha}) t_2 e^{t_2(x-U)} + e^{t_1 U}(-t_2)\frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha} \times \\ &\quad \frac{t_1 e^{t_1(x-U)} - t_2 e^{t_2(x-U)}}{t_1 - t_2}, \\ &= [(\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha}) t_2 + e^{t_1 U}(-t_2)\frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha}] \cdot e^{t_2(x-U)} + \\ &\quad e^{t_1 U}(-t_2 t_1)\frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha} \cdot \frac{e^{t_1(x-U)} - e^{t_2(x-U)}}{t_1 - t_2}. \end{aligned} \tag{5.44}$$

Now suppose we have

$$(\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha}) t_2 + e^{t_1 U}(-t_2)\frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha} > 0. \tag{5.45}$$

Then we readily obtain from (5.44) that

$$Y''(x) \geq 0, \quad \text{for all } x \in (U, 0).$$

Therefore, to show that Y is convex on $(U, 0)$, it remains for us to establish (5.45). We first note from (5.24) that

$$\begin{aligned} & t_2(\alpha\mathcal{D}_p + \mathcal{C}_-) + (\mathcal{C}_+ + \mathcal{C}_-)(-t_2) \cdot e^{t_1 U} \\ &= \quad (r_1 + r_2)(\alpha\mathcal{D}_p + \mathcal{C}_-) - [B_2 \frac{r_2}{r_1 - r_2} e^{r_1(U-L)} \\ &\qquad + J_2 \frac{r_1}{r_1 - r_2} e^{r_2(U-L)}], \end{aligned} \tag{5.46}$$

and thus (5.45) holds if and only if

$$(r_1 + r_2)K + r_2 B_2 e^{r_1(U-L)} + r_1 J_2 e^{r_2(U-L)} < 0,$$

which due to (5.23) is equivalent to

$$r_1 B_2 e^{r_1(U-L)} + r_2 J_2 e^{r_2(U-L)} > 0. \tag{5.47}$$

Define

$$f(x) = B_2 e^{r_1 x} + J_2 e^{r_2 x} + K.$$

We deduce that

$$f'(0) = r_1 B_2 + r_2 J_2 = (r_1 - r_2) s_1 (\mathcal{C}_- - \alpha\mathcal{I}_p) > 0. \tag{5.48}$$

One can readily verify that f is convex. Upon combining this with (5.48), we obtain

$$f'(U - L) > 0,$$

which is exactly (5.47). Finally, we show that for $x > 0$

$$Y''(x) = l_3 t_2^2 e^{t_2 x} \geq 0.$$

It suffices to show $l_3 \geq 0$, which by (5.43) is equivalent to showing

$$\lambda_2 \geq \frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha} (\frac{1}{t_2} + \frac{1}{t_1 - t_2}). \tag{5.49}$$

From (5.37) and (5.42), we know

$$\lambda_2 = (\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha}) \frac{e^{-t_2 U}}{t_2} - \frac{\sigma^2}{2\alpha^2} (\mathcal{C}_+ + \mathcal{C}_-) \frac{t_1 t_2}{t_1 - t_2} e^{(t_1 - t_2)U}.$$

Upon substituting this expression into (5.49) and multiplying both sides by $t_2(t_1-t_2)$, we simply need to show

$$(\mathcal{D}_p + \frac{\mathcal{C}_-}{\alpha})(t_1 - t_2)e^{-t_2 U} - \frac{\sigma^2}{2\alpha^2}(\mathcal{C}_+ + \mathcal{C}_-)t_1 t_2^2 e^{(t_1-t_2)U} \leq \frac{\mathcal{C}_+ + \mathcal{C}_-}{\alpha} t_1. \quad (5.50)$$

Given that $\mathcal{D}_p < \mathcal{C}_+/\alpha$ and using the fact that

$$t_1 t_2 = \frac{-2\alpha}{\sigma^2},$$

then establishing the inequality (5.50) is equivalent to showing [48]

$$e^{-t_2 U}(t_1 - t_2) + t_2 e^{(t_1-t_2)U} \leq t_1. \quad (5.51)$$

To this end, we set

$$g(y) = e^{-t_2 y}(t_1 - t_2) + t_2 e^{(t_1-t_2)y} - t_1,$$

and verify

$$g'(y) = -t_2(t_1 - t_2)e^{-t_2 y}(1 - e^{t_1 y}),$$

which implies that

$$g'(y) \geq 0 \qquad \text{for } y \leq 0.$$

Combining this with the fact that $g(0) = 0$, we deduce for $U \leq 0$

$$g(U) \leq g(0) = 0,$$

thus showing (5.51). We have therefore established the convexity of the candidate value function Y, which implies (5.32).

5.A.2.2 Uniqueness of the threshold values

Finally, we show that L and U are uniquely determined by two nonlinear equations and provide necessary and sufficient conditions under which the two equations are both negative, both positive, or of different signs. Once again, we focus here on

proving these results for Case II of Theorem 5.3.1, noting that Cases I and III and other technical details can be found in [48]. Since

$$\mathcal{I}_p + \mathcal{D}_p > 0,$$

we obtain

$$B_2 + J_2 + K = \alpha(r_1 - r_2) \cdot (-\mathcal{I}_p - \mathcal{D}_p) < 0.$$

Defining

$$f(x) = B_2 e^{r_1 x} + J_2 e^{r_2 x} + K,$$

one can readily verify that f is convex,

$$f(0) = B_2 + J_2 + K < 0$$

and

$$\lim_{|x| \to \infty} f(x) = \infty,$$

which implies that $f(x) = 0$ has only one positive solution. We therefore conclude that (5.23) uniquely determines $U - L$ and that (5.24) uniquely determines U. To establish necessary and sufficient conditions under which $U \leq 0$, observe that [48]

$$\begin{aligned}
U \leq 0 \quad &\Leftrightarrow \quad e^{t_1 U} \leq 1, \\
&\Leftrightarrow \quad \text{RHS (5.24)} \geq (r_1 + r_2 - t_2)(\alpha \mathcal{D}_p + \mathcal{C}_-) + \\
&\qquad\qquad\qquad (\mathcal{C}_+ + \mathcal{C}_-) t_2, \\
&\Leftrightarrow \quad \text{LHS (5.24)} \geq (r_1 + r_2 - t_2)(\alpha \mathcal{D}_p + \mathcal{C}_-) + \\
&\qquad\qquad\qquad (\mathcal{C}_+ + \mathcal{C}_-) t_2, \\
&\Leftrightarrow \quad e^{r_1 (U - L)} \leq \frac{B_3 - B_1}{B_2}.
\end{aligned}$$

Considering (5.23) and letting

$$h(y) = B_2 y + J_2 y^{\frac{r_2}{r_1}} + K$$

for $y > 0$, one can readily verify that h is strictly increasing on $[1, \infty)$ with $h(1) < 0$. Guaranteeing that (5.23) has a solution is equivalent to showing that $h(y) = 0$ has a solution in the interval

$$(1, \frac{B_3 - B_1}{B_2}].$$

Hence, it suffices to show

$$\frac{B_3 - B_1}{B_2} > 1 \quad \text{and} \quad h(\frac{B_3 - B_1}{B_2}) \geq 0, \tag{5.52}$$

which can be confirmed by simple algebraic manipulations establishing that the conditions (5.52) are the same as the second set of conditions in Theorem 5.3.1.

5.A.3 Proof of Theorem 5.3.1: Step 3

The final step of our proof of Theorem 5.3.1 consists of verifying that the proposed two-threshold dynamic control policy is optimal and that $Y(x) = V(x)$ for all x. We take a martingale argument approach where the key idea is to construct a submartingale to prove that the candidate value function Y is a lower bound for the optimization problem (5.9) - (5.14). To this end, first consider an admissible process $dP(t) = \theta dt$, where $P(t)$ is adapted to the filtration $\mathcal{F}_t$ generated by $\{D(s) : 0 \leq s \leq t\}$ and $\theta \in [\theta_\ell, \theta_u]$. Recalling $X(t) = P(t) - D(t)$, we define the process

$$\begin{aligned} M(t) = e^{-\alpha t} Y(X(t)) + \int_0^t e^{-\alpha s} (\mathcal{C}_+ X(s)^+ + \mathcal{C}_- X(s)^- \\ + \mathcal{I}_p \theta \mathbb{1}_{\{\theta \geq 0\}} - \mathcal{D}_p \theta \mathbb{1}_{\{\theta < 0\}}) ds, \end{aligned}$$

with our goal being to show that $M(t)$ is a submartingale.

Applying Ito's formula to $e^{-\alpha t} Y(X(t))$ renders for any $t_1 \leq t_2$

$$\begin{aligned} M(t_2) - M(t_1) = \int_{t_1}^{t_2} e^{-\alpha s} \Big(& -\alpha Y(X(s)) + \frac{1}{2}\sigma^2 Y''(X(s)) \\ & + (\theta - b) Y'(X(s)) + \mathcal{C}_+ X(s)^+ + \mathcal{C}_- X(s)^- \\ & + \mathcal{I}_p \theta \mathbb{1}_{\{\theta \geq 0\}} - \mathcal{D}_p \theta \mathbb{1}_{\{\theta < 0\}} \Big) ds \\ & - \int_{t_1}^{t_2} e^{-\alpha s} Y'(X(s)) \sigma dW(s). \end{aligned} \tag{5.53}$$

We have established in Section 5.A.2 that Y satisfies the Bellman equation

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) + \mathcal{C}_+ x^+ + \mathcal{C}_- x^- + \inf_{\theta_\ell \le \theta \le \theta_u} \mathcal{L}(\theta, x) = 0,$$

where

$$\mathcal{L}(\theta, x) = \begin{cases} (Y'(x) + \mathcal{I}_p)\theta & \text{if } \theta \ge 0, \\ (Y'(x) - \mathcal{D}_p)\theta & \text{if } \theta < 0. \end{cases}$$

This implies, for any given x and any $\theta \in [\theta_\ell, \theta_u]$, that

$$-\alpha Y(x) + \frac{1}{2}\sigma^2 Y''(x) - bY'(x) + \mathcal{C}_+ x^+ + \mathcal{C}_- x^- + \mathcal{L}(\theta, x) \ge 0.$$

Since $Y'(\cdot)$ is bounded, upon taking the conditional expectation in (5.53) with respect to the filtration $\mathcal{F}_{t_1}$, we deduce for any $t_1 \le t_2$ that

$$\mathbb{E}_x[M(t_2)|F_{t_1}] - M(t_1) \ge 0.$$

Namely, $M(t)$ is a submartingale and therefore we have

$$\mathbb{E}_x[M(t)] \ge M(0) = Y(x), \qquad \text{for any } t \ge 0.$$

Letting t go to ∞, we can conclude that Y is a lower bound for the optimal value of the optimization problem (5.9) - (5.14), and thus $Y(x) = V(x)$ for all x. Hence, the dynamic control policy characterized by the two threshold values L and U is indeed optimal, and our proof of Theorem 5.3.1 is complete.

Run-time Optimization of Composite Web Services with Response Time Commitments

6

In this chapter we address dynamic decision mechanisms for composite web services. We represent the composite web-service as a (sequential) workflow of tasks. For each task within this workflow, a number of third-party service alternatives may be available. We assume that the third-party service (task) alternatives offer the same functionality at different price-quality levels. Before a task in the workflow will be executed, a service alternative has to be selected that implements the task functionality. Decisions are represented in a decision strategy which defines decisions for all tasks in the workflow. Our goal is to find a dynamic strategy that maximizes the expected benefit for the composite service providers. For each task, the service-selection decision may be based on information about: observed response times in the current workflow, sub-service costs, response time characteristics of the alternatives, and end-to-end response time objectives with the corresponding rewards and violation penalties. We propose an approach, based on dynamic programming, to determine the optimal, dynamic selection policy. Numerical examples show significant potential gain in expected benefits using the dynamic approach compared to other, non-dynamic approaches.

6.1 Background

Composite web services in a service oriented architecture (SOA) aggregate web services that may be deployed and executed within different administrative domains. The composite web service provider typically runs an orchestrator that invokes the aggregated services according to a pre-defined workflow. The workflow is based on an unambiguous functionality description of a task ("abstract service"), and several alternatives ("concrete services") may exist that implement such a description [93]. With respect to functionality, all concrete services that match the same task service are identical. We refer to [42], and references therein, for an overview of QoS control frameworks.

A lot of attention in the literature has been paid to the problem of QoS-aware optimal service composition of SOA services (see, e.g. [26, 112, 113]). The main problem addressed in these papers is how to select one concrete service per task for a given workflow. This selection is made with the goal to guarantee the QoS of the composite service (as expressed in the respective SLA) while at the same time

[6] This chapter is based on [120], [122] and [121].

optimizing certain objectives like cost minimization. For the same QoS parameter, different SLAs are possible, ranging from those based on single value (e.g. expected value, median, etc.) to those models that are based on probabilistic models, represented by probability density functions, [117, 7, 122]. The choice of model impacts how QoS of composite services is determined, and consequently, how service selections are made. In static service composition solutions, the composition would remain unchanged the entire life-cycle of the composite web service. In static service composition solutions, the SLA violations could occur relatively often, leading to providers' losses and customer dissatisfaction. This is due to the high variability of the service environment, e.g. response time, and due to the fact that this is not well-reflected in static service composition solutions.

One way to address the problem of SLA violations is the request replication approach, as presented in [111]. Although showing relative potential, the request replication may be inefficient and not scalable compared to other approaches. It is suggested in [27, 114, 77] that, based on observations of the actually realized performance, re-composition of the service may be triggered, in order to mitigate the issue of SOA violations. These "reactive" solutions may select new concrete service(s) for the given workflow during the re-composition phase. Once the re-composition phase is over, the (new) composition is used as long as there are no further SLA violations. In particular, the authors of [27, 114, 77] describe *when* to trigger such (re-composition) event, and which adaptation actions may be used to improve overall performance. On the other hand, dynamic service compositions consider how to dynamically adapt the service composition at runtime [28].

It seems natural to address the issue of dynamic service composition within SOA by application of Markov Decision Processes (MDPs) and variants thereof. Abundo et al. [1] use MDPs to calculate the admission policy which allows the orchestrator to decide whether to accept or reject a new potential user in such a way to maximize the benefit while guaranteeing non-functional QoS for already admitted users. Wang et al. [106] use MDPs to model service composition with the aim to create automatically an abstract workflow of the service composition that satisfies functional and non-functional requirements, and also to allow the composite service to adapt dynamically to a varying environment. Yet another example of the pro-active solution for SOA-based systems based upon MDP has been considered in [92]. For a sequential workflow, a replacement of the atomic service is initiated when the maximum loss a consumer could bear for each service occurs. The authors do not consider service replacement at each SLA violation. The problem of an adaptive service composition is also addressed in [105], where a solution based on a hierarchical reinforcement learning method has been considered.

All of the mentioned MDP-based solutions show significant improvements over the respective non-dynamic (i.e. static) service composition solutions. However, relatively little is done with respect to a analysis of the services' QoS parameter space.

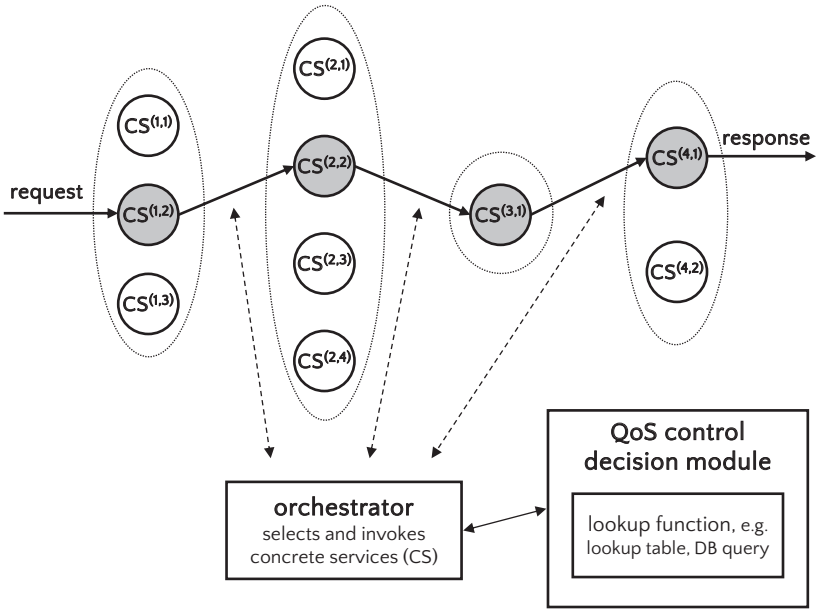


Figure 6.1: *Composite web service depicted by a sequential workflow. Dynamic service composition is based on pre-calculated response-time thresholds using dynamic programming.*

Motivated by this, in this chapter we conduct a thorough empirical evaluation based on our analysis from [122]. In [122] we analyzed our dynamic service composition approach that is based on DP. The objective is to maximize benefit for the composite service provider, while committing to the response-time objective that is part of the agreed end-to-end SLA. The investigation in this chapter aims to fully explore the impact of the reward, penalty and execution cost parameters, relating to the expected value and the variance of the response-time probabilistic models (distributions). To the best of our knowledge, the impact of the number of the concrete services (alternatives) and the positions of the alternatives within the workflow has not been analyzed so far within the context of the MDP-based SOA dynamic service composition. We quantify the answer to the question how many alternatives are advisable at the "beginning", "middle" and "end" of a workflow, and enlight this analysis with respect to the number of services within the workflow.

After illustration of MDP-based solution in Section 6.2 we describe in Section 6.3 the model of the system under consideration. In Section 6.4 we describe the DP approach we abridged for our purposes. We explain the procedure and derive formu-

lae used to determine response time thresholds. Simulation results are presented and discussed in Section 6.5 in which we illustrate the influence of different system parameters. In Section 6.6 we conclude this chapter with directions for further research.

6.2 Motivating example

To illustrate our dynamic service selection approach, we consider the case of sequential workflows. Figure 6.1 depicts a sequential workflow consisting of four tasks (defined as abstract services). Each task maps to a number of concrete services (alternatives). Per task, the number of alternatives is three, four, one and two, respectively. The response times of the concrete web services are represented by random variables for with it is assumed that the probability distributions are known in advance. See Remark 6.3.1 for when this is not the case. The execution costs per concrete service are known in advance as well. Our approach is tailored for sequential workflows. However, our approach is applicable to any workflow that could be aggregated and mapped into a sequential one. This can be done by the calculations of the aggregated services for the most frequently used workflow patterns in which the probabilistic models are used as explained in [122], [117], and [7]. In case of arbitrary workflow patterns, an efficient numerical method could be used, as presented in [34]. See Appendix 6.A for an example how where an (non sequantial) example workflow is aggegrated into a sequential one.

After each execution of a single task within the workflow, the orchestrator decides for the next task which concrete service alternative will be executed. To this end the orchestrator compares the remaining end-to-end deadline time-budget to the pre-calculated response-time thresholds for each service. The thresholds are retrieved from a pre-calculated lookup table. In the most extreme example, after the execution of, e.g. the first task within the workflow, the promised end-to-end deadline may already has been violated. In such a case, the dynamic selection algorithm should choose the cheapest alternatives for all remaining tasks in the workflow. Similarly, when only the last task remains to be executed, and the promised deadline is jeopardized, it may be sensible to select the more expensive service with better deadline-meeting promise than the cheaper service with the much smaller probability of meeting the deadline.

When the client's request meets the agreed end-to-end deadline, the composite service provider is rewarded by the client. Otherwise, when the deadline is not met (i.e. an event of an SLA violation) the provider pays a penalty to the client. The exact relation between the metric of percentile conformance and the monetary penalties charged on the provider due to non-conformance (hereon referred to as the "penalty function") is an important parameter of the SLA [16]. There are multiple ways in

relating the two and thus a variety of penalty functions can be chosen. We have adopted the step-wise penalty function in this case with the desired conforming percentile of 100% [16].

The response-time thresholds are calculated before the first request is admitted to the system, where the DP takes the penalty, the reward, the concrete services' level objectives (SLOs) and the execution costs as input. These thresholds are represented by e.g. a lookup table. Depending upon the actual response times and the thresholds, it is possible that each client request may be served by a different chain of alternatives.

6.3 Model

A composite service consists of N of tasks or *abstract services*, denoted by $T_1, \dots, T_N$, that have to be performed in sequential order. For each task i there are M_i alternative implementations available, which are called *concrete services*, denoted by $CS^{(i,j)}$, for $j = 1, \dots, M_i$. The composite service processes incoming service requests. To this end, each request is assigned a unique concrete service alternative for each task in the composition chain. In other words, each request is assigned a *workflow of concrete services* $CS^{(1,w_1)} \rightarrow \cdots \rightarrow CS^{(N,w_N)}$ that contains for each task T_i an invocation of a concrete service alternative $CS^{(i,w_i)}$. Each workflow is represented in a workflow vector $\boldsymbol{W}$, defined as:

$$\boldsymbol{W} := [w_1, \dots, w_N]. \tag{6.1}$$

The position of w_i in the vector corresponds to the position in the workflow and the values $w_i = 1, \dots, M_i$ correspond to the concrete service alternative at position i.

Figure 6.2 illustrates the case where $N = 4$, $M_1 = 3$, $M_2 = 4$, $M_3 = 1$, $M_4 = 2$ and $\boldsymbol{W} = [2, 2, 1, 1]$.

Per single composite service request, the orchestrator (illustrated in Figure 6.1) executes services one-by-one as indicated by the workflow. Before task T_i is executed, the orchestrator makes a decision which one of the M_i service alternatives to choose. Decisions from the orchestrator are based on information about response times from the concrete services. The response-time SLOs of the concrete services are specified as "soft" ones, and in general, "soft" SLOs are expressed as a response-time probability distribution function (PDF) [96], or alternatively, as the cumulative distribution function (CDF).

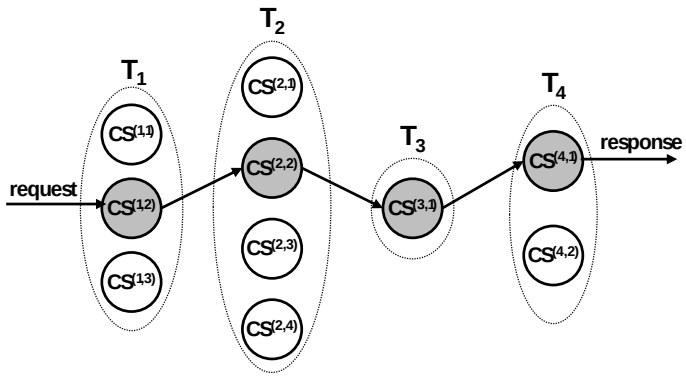


Figure 6.2: *Model with example workflow* $CS^{(1,2)} \rightarrow CS^{(2,2)} \rightarrow CS^{3,1} \rightarrow CS^{(4,1)}$*, corresponding workflow vector* $\boldsymbol{W} = [2, 2, 1, 1]$ *and* $M_1 = 3,\ M_2 = 4,\ M_3 = 1,\ M_4 = 2$.

6.3.1. Remark (probability density functions)**.** In practice, probability density functions may either be estimated from the measurements carried out by the composite service provider, or the third-party domains may publish, or otherwise make available, such information. Since we investigate the potential of our approach, we assume time-invariant SLAs and the PDFs (which are part of the SLAs) do not change either. In case of time-variant PDFs, a recalculation of the response-time thresholds may be occasionally necessary. The recalculation may be triggered when there is a long-term SLA violation, i.e. when the observed PDF differs "significantly" from the initial one. This is topic is addressed in Chapter 7.

Each concrete service $CS^{(i,j)}$ (the j-th alternative for task i) has a response time represented by a random variable $D^{(i,j)} \geq 0$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M_i$. From the perspective of the response time, we model each concrete service as a black box, which means that for each random variable $D^{(i,j)}$ the respective PDF (or CDF) is given. The PDFs and CDFs for concrete services are denoted by $f^{(i,j)}(t)$ and $F^{(i,j)}(t)$, respectively. If a concrete service $CS^{(i,j)}$ is invoked a response time realization $d^{(i,j)}$ ($i = 1, \ldots, N,\ j = 1, \ldots, M_i$) is generated and a invocation cost of $c^{(i,j)}$ money units have to be paid. Because $CS^{(i,j)}$ is represented by random variable $D^{(i,j)}$ with given distribution, $d^{(i,j)}$ is a sample from the distribution associated with $D^{(i,j)}$. We assume that response times of concrete service alternatives are mutually independent. Using the mutual independence, for a given workflow $CS^{(1,w_1)} \rightarrow \cdots \rightarrow CS^{(N,w_N)}$ the response time distribution can be determined by taking the convolution of the distributions associated to the respective random variables:

$$D_{\boldsymbol{W}} = D^{(1,w_1)} \star D^{(2,w_2)} \star \cdots \star D^{(N,w_N)}. \tag{6.2}$$

The realization of a end-to-end response time, resulting from invoking workflow $\boldsymbol{W}$, is represented by $D_{\boldsymbol{W}}$. The overall deadline for a request handled by the orchestrator is denoted by δ_p. For each workflow $\boldsymbol{W}$ the probability of a successful response within δ_p is defined by:

$$p_{\boldsymbol{W}} := \mathbb{P}(D_{\boldsymbol{W}} \leq \delta_p). \tag{6.3}$$

The workflow invocation cost $c_{\boldsymbol{W}}$ for workflow $\boldsymbol{W} = (w_1, \ldots, w_N)$ is defined by:

$$c_{\boldsymbol{W}} := \sum_{i=1}^{N} c^{(i,w_i)}. \tag{6.4}$$

As for a fixed workflow $\boldsymbol{W}$ the composition is known, an explicit expression for the expected benefit per request can be formulated. We define $R^*_{\boldsymbol{W}}$ as the expected benefit per request for workflow $\boldsymbol{W}$:

$$R^*_{\boldsymbol{W}} := Rp_{\boldsymbol{W}} - V(1 - p_{\boldsymbol{W}}) - c_{\boldsymbol{W}}. \tag{6.5}$$

When decisions are made in a dynamic fashion, the concrete service workflow $\boldsymbol{W}$ is not fixed, but depends on the response time realizations of concrete services. In our approach the optimization problem is solved by applying DP. The key idea behind DP is that the optimization problem can be separated into smaller subset problems that are easier to solve. Solving the smaller subset problems will lead to the solution of the optimization problem that optimizes expected benefit. Dynamic programming consists of the following components:

- **State space** $\mathcal{S}$, in our case the state space is defined by $\mathcal{S} = \{1, \cdots, N\} \times \mathbb{R}^+$. Each state is a tuple $(i, b) \in \mathcal{S},\ (i \in \{1, \cdots, N\},\ b \in \mathbb{R}^+)$ combining position i in the workflow and remaining time until deadline violation b.
- **Decision epochs**, before execution of each next sub-service we have to select an concrete service alternative based on b.
- **Action space**, the set of possible actions for given state (i, b). At each decision moment at task T_i we have a set of concrete service alternatives $\{CS^{(i,1)}, \ldots, CS^{(i,M_i)}\}$, $i = 1, \ldots, N$.
- **Cost function**, defining cost for each action or state that is reached. If concrete service alternative $CS^{(i,j)}$ than invocation cost $c^{(i,j)}$ has to be paid. At the end of the workflow a reward R is earned is the deadline is not violated. Otherwise a penalty V has to be paid.
- **Value function** $P^{(i,*)}(b)$, representing the expected benefit for state $(i, b) \in \mathcal{S}$.

- **Policy** A, the set of decisions or actions over all states (i, b). The policy can be implemented by a lookup table where for given (i, b) the optimal pre-calculated decision is selected. An example is depicted in Figure 6.3 at Section 6.4. We refer to Section 6.4 for more details.

The combination of state space, decision epochs, action space, cost function and value function defines a DP problem where the solution results in policy that optimizes expected benefit. An optimal policy can be determined by defining a backward recursion on $P^{(i,*)}(b)$. This recursion is called the Bellman Equation [11]. The recursion is defined and implemented in Section 6.4. We denote the dynamic workflow that implements lookup table A as:

$$\boldsymbol{W}^d(A).$$

Generally there is no tractable explicit expression for the end-to-end response time distribution when the orchestrator chooses workflows dynamically like for the static workflow. For the orchestrator the realization of a end-to-end response-time distribution is denoted by D and has corresponding realization d. The fraction of requests within the deadline is

$$p := \mathbb{P}(D \leq \delta_p). \tag{6.6}$$

The orchestrator has to deal with two levels of SLAs:

First, the SLA agreed between the individual service provider (ISP) for the j-th service alternative of the i-th task and the composite service provider (CSP). These consist of the following elements:

- The response-time probability distribution function, $f^{(i,j)}(b)$.
- The cost $c^{(i,j)}$ [money unit] that the CSP pays to ISP for the execution of a single request. No penalties are imposed. From the ISP viewpoint, this value represents reward.

Second, the SLA between the CSP and its clients, that contains the following elements:

- The end-to-end response time penalty deadline δ_p [time unit].
- The fraction of response time realizations p_{e2e} that should be within the deadline δ_p.
- The reward R [money unit] that the CSP gets for executing a single request within penalty deadline δ_p.
- The penalty V [money unit] that the CSP pays to the end customer when the agreed end-to-end deadline is not met.

The orchestrator must choose workflows such that it will meet the required fraction p_{e2e} of requests within deadline δ_p:

$$\mathbb{P}(D \leq \delta_p) \geq p_{e2e}. \tag{6.7}$$

6.4 Algorithm description

In this section we describe how to optimize expected CSP benefit by formulating the dynamic service selection as a DP, [11]. Before a request is executed by the CSP a response time budget δ_p is available until response-time deadline δ_p is violated. Each execution of concrete service $CS_i(j)$ in the CSP workflow will result in a response-time realization $d^{(i,j)}$ and a remaining response-time budget B. After execution of $CS^{(i,j)}$, the remaining time budget B will reduce with $d^{(i,j)}$ time units. The term "dynamic" refers to the fact that workflows are selected on-line based on B, the remaining response-time budget until the deadline δ_p is violated. In other words, before the execution of each task, a service alternative must be selected, based on B. The DP optimizes the CSP benefit by taking into account the effect of future decisions. Since the decisions within the DP take in account (possible) effects subsequent decisions, a "backward recursion" method is needed for finding the optimal solution. The application of DP will result in a decision policy. The policy indicates, for given response time realizations, which one of the concrete service alternatives should be chosen in order to optimize the CSP's expected benefit per composite service request. The policy is determined by the current position within the sequential workflow i and the remaining time b ("time budget") till the overall deadline δ_p will be violated.

Define $\overline{P}^{(i,*)}(b) := \mathbb{E}[\text{Benefit} \mid B = b, I = i]$ as the expected benefit under the optimal DP policy. Here $B = b$ is the given response time budget , and i represents the task at the ith position in the workflow. The recursion starts with the terminal reward function for $b \geq 0$:

$$\overline{P}^{(N+1,*)}(b) = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{6.8a}$$

Using this function, we iterate backwards using the following equations, for $i = 1, \ldots, N, \quad j = 1, \ldots, M_i, \quad b > 0$:

$$\overline{P}^{(i,*)}(b) = \max_j \Big\{ -c^{(i,j)} + \overline{R}^{(i,j)}(b) + \overline{V}^{(i,j)}(b) \Big\}, \tag{6.8b}$$

$$\overline{R}^{(i,j)}(b) = \int_0^b f^{(i,j)}(t)\overline{P}^{(i+1,*)}(t-b)\mathrm{d}t, \quad \text{and} \tag{6.8c}$$

$$\overline{V}^{(i,j)}(b) = \int_b^\infty f^{(i,j)}(t)\overline{P}^{(i+1,*)}(0)\mathrm{d}t. \tag{6.8d}$$

Here $f^{(i,j)}(t)$ represents the response-time PDF of concrete service alternative j for task i, while the term $\overline{R}^{(i,j)}(b)$ represents the expected reward, when concrete service j (corresponding to task i) is executed for the given time-budget value $B = b$. The term $\overline{V}^{(i,j)}(b)$ represents the expected penalty for exceeding the overall deadline at task i while executing concrete service j for the given time budget value $B = b$. The expected reward and penalty functions take into account the impact of future decisions as represented by terms relating to $\overline{R}^{(i+1,*)}(b)$ in equations (6.8c) and (6.8d). Once the end of the workflow is reached (i.e., $i = N$) the current request has been processed.

The integrals in Equation (6.8c) will generally not result in tractable expressions. The problem can be solved numerically, by discretizing the distributions. For the discretization we split the time interval over which the response-time PDF is defined in segments of the same size h. The number of segments is T^* and the size of h corresponds to the accuracy of the discretization. The discretized versions of the PDF $q_k^{(i,j)}$ are therefore defined as follows, for $i = 1, \ldots, N,\ j = 1, \ldots, M_i,\ b = 0, \ldots, T^*$:

$$T^* = \left\lceil \frac{\delta^*}{h} \right\rceil,$$
$$q_b^{(i,j)} = \mathbb{P}\big(D^{(i,j)} \leq h[b+0.5]\big) - \mathbb{P}\big(D^{(i,j)} \leq h[b-0.5]\big),$$

Generally, the larger the number of segments T^* the more accurate the discretization would be, but it would take longer time to calculate the respective PDF.

Using the discretization, the backward recursion can be transformed into a scheme that can be evaluated numerically. For given number of segments T^*, let the terms

$P_b^{(i,*)}$, $R_b^{(i,j)}$, and $V_b^{(i,j)}$ represent discretized versions of $\overline{P}^{(i,*)}(b)$, $\overline{R}^{(i,j)}(b)$, and $\overline{V}^{(i,j)}(b)$, respectively. The backward recursion formulae are then as follows:

$$P_b^{(N+1,*)} = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{6.9a}$$

Using this function, we iterate backwards using the following equations, for $i = 1, \ldots, N, \quad j = 1, \ldots, M_i, \quad b = 0, \ldots, T^*$:

$$P_b^{(i,*)} = \max_j \left\{ -c^{(i,j)} + R_b^{(i,j)} + V_b^{(i,j)} \right\}, \tag{6.9b}$$

$$R_b^{(i,j)} = \sum_{k=0}^{b} q_k^{(i,j)} P_{k-b}^{(i+1,*)}, \quad \text{and} \tag{6.9c}$$

$$V_b^{(i,j)} = \sum_{k=b+1}^{T^*} q_k^{(i,j)} P_0^{(i+1,*)}. \tag{6.9d}$$

While applying formulae (6.9b)-(6.9d), the corresponding decisions (actions) A can be obtained by storing the maximum arguments evaluated for $i = 1, \ldots, N, \quad b = 0, \ldots, T^*$ as:

$$A_b^{(i,*)} = \operatorname*{argmax}_{j=1,\ldots,M_i} \left\{ -c^{(i,j)} + R_b^{(i,j)} - V_b^{(i,j)} \right\}.$$

The optimal decisions could be represented by a lookup-table, and a graphical example of a lookup-table for the sequential workflow with $N = 4$ tasks is shown in Figure 6.3. The horizontal axis corresponds to the time budget left until the overall deadline is breached, while the vertical axis corresponds to the position of the task within the workflow. The color corresponds to the decision that has to be taken, e.g. proceed with the alternative j.

We illustrate the lookup table with the following examples:

(1) We start handling a new request and the overall deadline equals $\delta_p = 13.5$. The decision is marked by an asterisk $*$ at the lookup table shown in Figure 6.3, and alternative 1 is to be selected.

(2) We have 4.9 time units remaining the decision is made for the task at position 3. The decision is marked by a cross $\times$, and points that service alternative 3 should be selected.

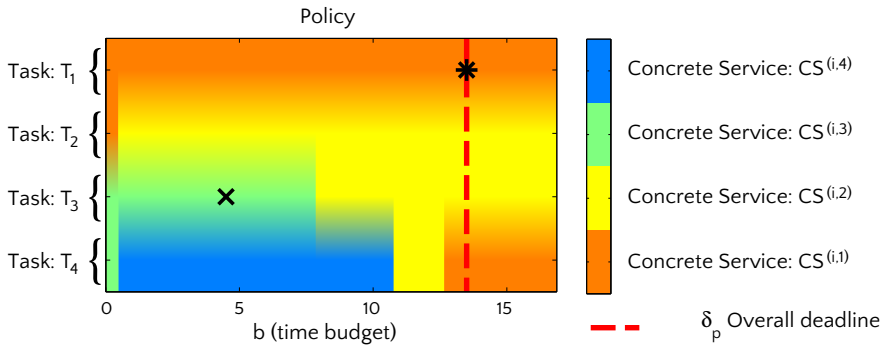


Figure 6.3: *Graphical example of a decision table.*

6.5 Numerical experiments

In this section we investigate the influence of the system parameters on the potential gain that can be obtained by applying dynamic service composition as compared to static service composition. To this end, we have performed a wide variety of numerical experiments. The results of these experiments are outlined below. The

Parameter	Definition
N	Number of tasks (workflow size)
M_i	Number of service alternatives for task i
$f^{(i,j)}$	PDF of response-time distribution $CS^{(i,j)}$
$\mu_{i,j}$	Mean response time of $CS^{(i,j)}$
$\sigma^2_{i,j}$	Variance of response time of $CS^{(i,j)}$
$c^{(i,j)}$	Cost of invocation of $CS^{(i,j)}$
δ_p	End-to-end deadline
p_{e2e}	Required fraction of responses within the deadline
R	Reward per successful request within deadline δ_p
V	Penalty per request not completed within deadline
K_R	Scaled reward
K_V	Scaled penalty

Table 6.1: *Overview of model parameters.*

parameters of the models discussed in Sections 6.3 and 6.4, and their corresponding value ranges, are listed in Table 6.1. The parameter space is highly dimensional and prohibits an exhaustive analysis of model instances. Therefore, we have defined a number of specific model instances that represent certain characteristics of service compositions. First, we study the impact of the input parameters reward, penalty

and cost on the potential gain in expected benefit obtained by dynamic versus static service composition. Second, we investigate the impact of the service-chain length on the potential gain. Third, we consider the impact of the number of available alternatives for the tasks, and their corresponding positions in the service chain, on the expected gain. For convenience, we assume that the response-time distributions for the j-th alternative of task i can be approximated by a log-normal distribution with mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$, see Table 6.1. We emphasize that this assumption is not restrictive, because our approach supports *all* non-negative probability distributions.

The *gain* G in expected benefit obtained by using optimal dynamic composition compared to the optimal static composition is defined as follows:

$$G := \frac{R^*_{dynamic} - R^*_{static}}{R^*_{static}} \times 100\%, \tag{6.10}$$

where $R^*_{dynamic}$ is the expected benefit per request under the optimal dynamic policy, obtained by applying the algorithm described in 6.4. Moreover, R^*_{static} is the expected benefit per request under the optimal static policy, obtained by an exhaustive search of all possible "paths" traversing the tasks in the workflow (see for example Figure 6.2, where there are 24 possible paths).

For our parameter study we need to choose a deadline δ_p. This has to be done such that experiments will generate sensible results. Therefore, we introduce the reference workflow $\boldsymbol{W}_{ref}$. Using the reference workflow we can relate a sensible deadline to a given end-to-end objective p_{e2e}. The reference workflow $\boldsymbol{W}_{ref}$ is determined by taking

$$\boldsymbol{W}_{ref} = \operatorname*{argmax}_{\boldsymbol{W}} \left[R p_{\boldsymbol{W}} - V(1 - p_{\boldsymbol{W}}) - c_{\boldsymbol{W}} \right], \tag{6.11}$$

$$c_{\boldsymbol{W}} := \sum_{i=1}^{N} c^{(i,w_i)}, \tag{6.12}$$

$$p_{\boldsymbol{W}} := \mathbb{P}(D_{\boldsymbol{W}} \leq \delta_p). \tag{6.13}$$

For given p_{e2e}, we denote by δ_p the deadline such that for the reference composition $\boldsymbol{W}_{ref}$, consisting of $CS^{(i,1)}$, $i = 1, \ldots, N$, it holds that $\mathbb{P}(D_{ref} < \delta_p) = p_{e2e}$.

6.5.1 Impact of reward, penalty and cost parameters

In this subsection we study the impact of concrete service-cost $c^{(i,j)}$, the reward R and the penalty V on the potential gain G, defined in (6.10). To this end, define the overall mean service cost by

$$\overline{c} = \sum_{i=1}^{N} \overline{c}_i, \quad \text{where} \quad \overline{c}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} c^{(i,j)}. \tag{6.14}$$

Note that R, V and $\overline{c}$ are scale-invariant, in the sense that if these parameters are scaled to αR, αV and $\alpha \overline{c}$ for some $\alpha > 0$, then the optimal dynamic and static policies, and hence also the gain G, remain the same. Therefore, it is convenient to define the following two scaled reward and cost parameters:

$$K_R := \frac{R}{\overline{c}} \quad \text{and} \quad K_V := \frac{V}{\overline{c}}. \tag{6.15}$$

As an illustration, consider the following exploratory example with $N = 2$, $M_1 = 1$ and $M_2 = 2$, depicted in Figure 6.4, and where the cost parameters $c^{(i,j)}$ and the distribution parameters $\mu_{i,j}$ and $\sigma^2_{i,j}$ are listed in Table 6.2.

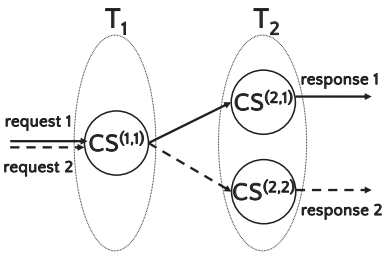


Figure 6.4: *Exploratory example with $M_1 = 1,\ M_2 = 2$.*

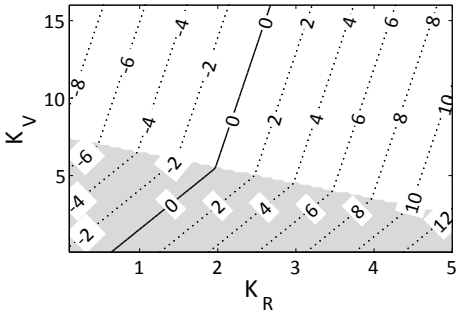
		Task $(i, \cdot)$					
		$(\mathbf{1}, \cdot)$			$(\mathbf{2}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4	1	5	4
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	5	2.5	4

Table 6.2: *Global scenario.*

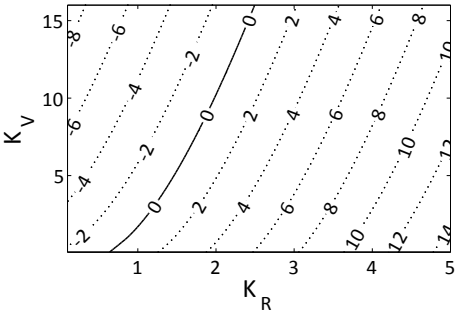
Figure 6.5a shows the contour lines of combinations (K_R, K_V) that lead to the same expected benefit, subject to the constraint that the reference *static* workflow (i.e.,

the workflow $[1, 1]$) leads to an end-to-end response time less than the deadline with probability $p_{e2e} = 80\%$. The values plotted on the dotted lined indicate the expected benefit. The grey area is the set of combinations for which the static workflow is $[1, 1]$, indicated by the solid line in Figure 6.4. The white area represents the combinations for which the static workflow $[1, 2]$ is optimal, indicated by the dashed line in Figure 6.4. Note that in these cases $p_{e2e} > 80\%$, because one chooses a faster alternative while maintaining the same deadline.

(a) *Expected benefit in static set-up.*



(b) *Expected benefit in dynamic set-up.*



(c) *Potential gain in expected benefit.*

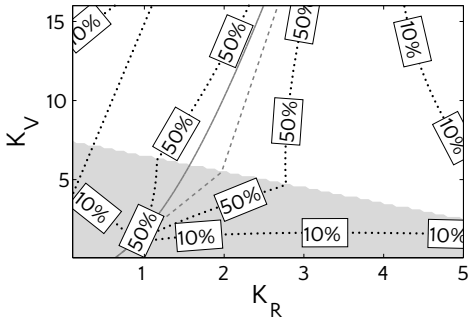


Figure 6.5: *Comparison between static and dynamic set-up for $p_{e2e} = 80\%$. Black contour lines represent equal values. The grey and white area correspond to the different optimal static compositions. Furthermore, the solid grey contour line is where $R^*_{dynamic} = 0$, the dashed grey contour line is where $R^*_{static} = 0$.*

Note that all iso-curves in Figure 6.5a are piecewise linear. More precisely, one may verify using (6.5) that for a given optimal static workflow the iso-curves are of the form $K_V = aK_R + b$, with

$$a = \frac{p_{\boldsymbol{W}}}{1 - p_{\boldsymbol{W}}} \quad \text{and} \quad b = \frac{R^*_{static} + c_{\boldsymbol{W}}}{\overline{c}(p_{\boldsymbol{W}} - 1)}, \tag{6.16}$$

where R^*_{static} is defined in (6.10), and where $c_{\boldsymbol{W}}$ is the cost for the optimal static workflow $\boldsymbol{W}$, $p_{\boldsymbol{W}}$ is probability a request using $\boldsymbol{W}$ will be within deadline δ_p and $\boldsymbol{W}$ is the optimal static workflow for which holds that $\boldsymbol{W} = [1, 1]$ in the grey area and $\boldsymbol{W} = [1, 2]$ in the white area. Moreover, it is readily verified from (6.5) that the switching curve of the optimal workflow (i.e., the border separating the grey and the white area) is given by:

$$K_V + K_R = \frac{c_{[1,1]} - c_{[1,2]}}{\overline{c}(p_{[1,1]} - p_{[1,2]})}, \tag{6.17}$$

where $c_{[1,k]}$ is the cost for workflow $[1, k]$ $(k = 1, 2)$ as defined in (6.4), $\overline{c}$ is defined in (6.14) and $p_{[1,k]}$ is the probability that the response time meets the deadline if the static workflow $[1, k]$ is optimal.

Figure 6.5b shows the results for the dynamic counterpart of Figure 6.5a, and Figure 6.5c shows the iso-curves for the relative gain G, defined in (6.10). Most remarkably, the iso-curves depicted in Figure 6.5c are wedge-shaped with a sharp angle at the switching curve (6.18). This suggests that the relative gain G has the highest values at the switching curve. In other words, *the information about the response times at the first task is particularly crucial to decide about the optimal path.*

The solid grey line is the contour line over the combinations (K_V, K_R) where $R^*_{dynamic} = 0$, and the dashed grey line show the combinations for which $R^*_{static} = 0$. We observe that the distance between these two lines is maximal on the switching curve, as expected. Moreover, note that the green line gives the boundary where positive benefit is expected: combinations to the left of this curve will result in loss in expectation.

Figures 6.6a to 6.6c show similar results to those in Figures 6.5a to 6.5c, but with $p_{e2e} = 90\%$.

6.5.2 Impact of number of alternatives

We will now investigate the impact of the number of available alternatives for each of the tasks on the gain G to be obtained by using dynamic composition. To this end, we consider a service chain of length $N \geq 3$, and with $M_i = 1$ for $i \neq 2, N$, and

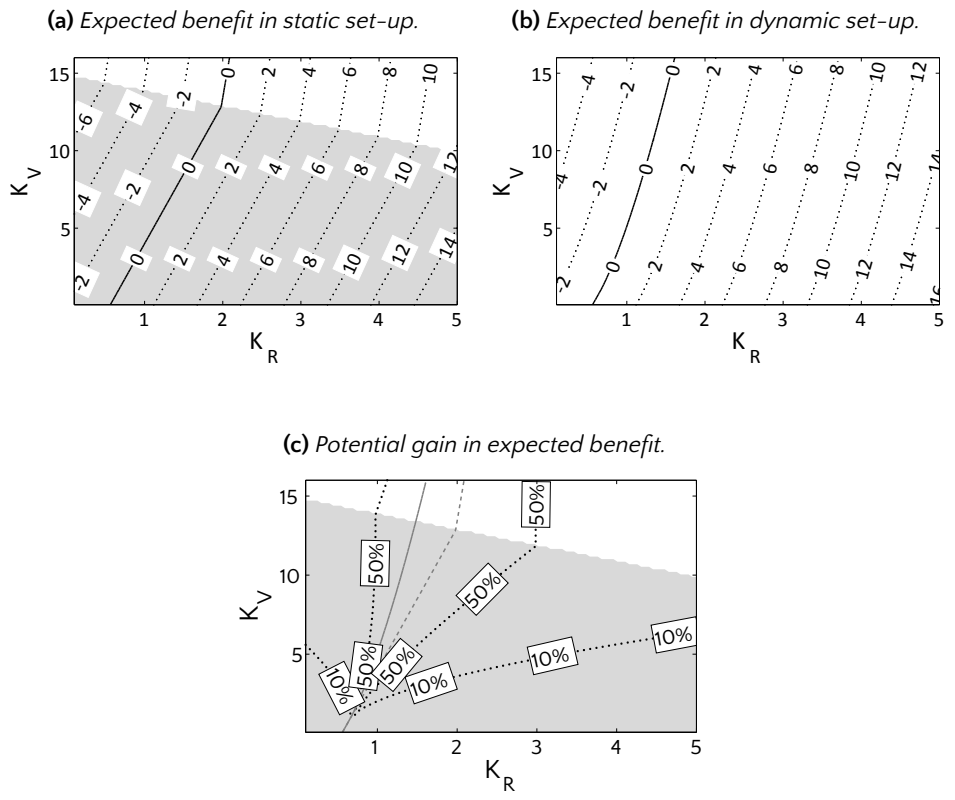


Figure 6.6: *Comparison between static and dynamic set-up for $p_{e2e} = 90\%$. Black contour lines represent equal values. The grey and white area correspond to the different optimal static compositions. Furthermore, the solid grey contour line is where $R^*_{dynamic} = 0$, the dashed grey contour line is where $R^*_{static} = 0$.*

where M_2 and M_N are varied as $\{1, 2, 3, 4\}$. The cost and distribution parameters are listed in Table 6.3, and moreover, we take $R = 26$, and $V = 130$. Note that it is readily verified from (6.14) that $\overline{c} = 26$, and from (6.15) that $K_R = 1$ and $K_V = 5$.

Figure 6.7 illustrates an example with $M_2 = 2$ and $M_N = 3$. For each of the model instances, we have calculated (1) the optimal static composition and its corresponding expected benefit R^*_{static}, (2) the optimal dynamic composition and its corresponding expected benefit $R^*_{dynamic}$, and (3) the relative gain G, defined in (6.10).

In particular we expect that, due to increasing uncertainty when traversing a service composition workflow, more alternatives are desired at the end of the workflow. For our analyses we define a service chain of length N where at the second position and at the last position the number of concrete service alternatives is varied. With this

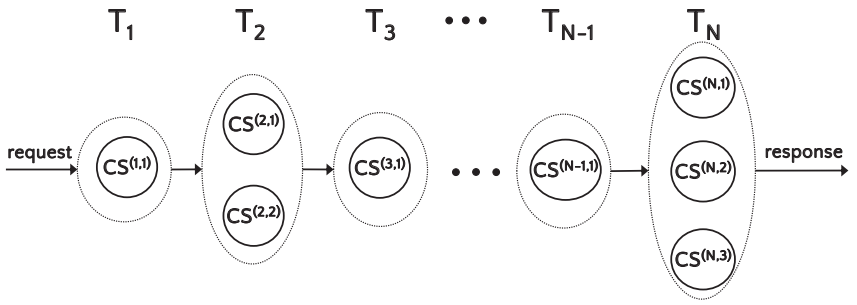


Figure 6.7: *Configuration for analyses on impact of position of alternatives at task T_2 at position 2 and task T_N at position N. In this case $M_2 = 2$, $M_N = 3$ and $M_i = 1$ for $i \in \{1, \ldots, N\} \setminus \{2, N\}$.*

set-up we can compare the relative value of alternatives in the beginning and at the end of the service composition chain. The results are presented in tables where the rows correspond to M_N and columns correspond to M_2. In other words, the rows correspond to the number of concrete service alternatives at position N and the columns correspond to the number of available concrete service alternatives at position 2. The parameter values are presented in Table 6.3. Results of the case where $N = 8$ and $p_{e2e} = 90\%$ for the reference composition are represented in Table 6.4. We observe that increasing the number of alternatives at position 2 in the workflow results in a neglectable increase in gain G. However increasing the number of alternatives at position 8 results in a increase in gain up to 42.1%. If we take a closer look to the expected benefit of the static composition approach in Table 6.4 we observe that the third and fourth alternative are never used as the expected benefit does not increase when the third and fourth alternative are available. Furthermore, we observe that the table is symmetric for the number of alternatives at positions 2 and 8. This is caused by the fact that workflows in the static scenario are fixed and therefore, if the alternatives at both positions are similar, the position where the number alternatives is increased has no effect on expected benefit. However, for the dynamic composition the position where the number of alternatives is increases has a dramatic effect. At the expected dynamic benefit results in Table 6.4, for the case with dynamic workflows, we observe that for position 8 the availability of alternatives 3 and 4 has a significant impact on benefit. At position 2 the availability has hardly any effect as the advantage of optimal dynamic selection is averaged out over the successive workflow response times. At position 8 the dynamic workflow enables the orchestrator to take advantage of the 3th and 4th alternative where the static workflow is not optimally using the 2th alternative and does not even use the 3th and 4th alternative.

		Task $(i, \cdot)$					
		$\mathbf{i} \notin \{\mathbf{2}, \mathbf{N}\}$			$\mathbf{i} \in \{\mathbf{2}, \mathbf{N}\}$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	16	1	5	16
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	3	2.5	4
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	9	1.25	1
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	27	0.675	0.25

Table 6.3: *Cost and distribution parameters.*

Gain		N_2			
		1	**2**	**3**	**4**
N_8	**1**	0.0%	0.0%	0.4%	0.4%
	2	16.6%	14.6%	14.8%	14.8%
	3	37.1%	25.1%	25.3%	25.3%
	4	42.1%	27.6%	27.8%	27.8%

Dynamic benefit		N_2			
		1	2	3	4
N_8	**1**	2.44	5.43	5.45	5.45
	2	6.33	8.21	8.23	8.23
	3	7.44	8.97	8.98	8.98
	4	7.71	9.15	9.16	9.16

Static benefit		N_2			
		1	2	3	4
N_8	**1**	2.44	5.43	5.43	5.43
	2	5.43	7.17	7.17	7.17
	3	5.43	7.17	7.17	7.17
	4	5.43	7.17	7.17	7.17

Table 6.4: *Impact on availability of alternatives.*

6.5.3 Length of the composition

In section 6.5.2 we observed that the effect of alternatives at the start of the workflow can vanish due to averaging of response times over the succeeding services. We will now investigate the effect of service-chain length. The service chain used for the experiments is depicted in Figure 6.8 and has length N. There is one alternative at position 1 ($M_1 = 1$) while there are two alternatives at position $i \geq 2$ ($M_i = 2$ for $i = 2, \ldots, N$). The parameter space is large, therefore we limit us to workflows where there are two concrete-service alternatives for each task. Concrete-service

		Task $(i,\cdot)$					
		$(\mathbf{1},\cdot)$			$(\mathbf{2},\cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service	$(\cdot,\mathbf{1})$	1	5	16	1	5	16
alternative $(\cdot,j)$	$(\cdot,\mathbf{2})$	n.a.	n.a.	n.a.	3	2.5	4

Table 6.5: *Cost and distribution parameters for the impact of the service-chain length.*

alternative parameters are defined in Table 6.5. Reward parameter R, penalty parameter V are related to overall mean service cost $\overline{c}$ as is defined in (6.14) using scaled reward and penalty K_R and K_V which are defined in (6.15). Deadline δ_p is chosen as defined in (6.11). Note that there are N different static workflows as for the static workflow the position of alternatives does not matter (the set of alternatives is equal for all tasks). We define a workflow for a system with N tasks as $\boldsymbol{W}_k$, $k = 0, \dots, N-1$ where k represents the number of faster alternatives that is invoked in the workflow. In our case the reference workflow becomes $\boldsymbol{W}_{ref} = \boldsymbol{W}_0 = [1, 1, \dots, 1]$ (a workflow consisting of only regular (slow) alternatives). We chose to vary parameters K_R and K_V in the ranges $K_R \in (0; 5]$ and $K_V \in (0; 50]$. Figures 6.9-6.13 contain results for service-chain length $N = 3, 6, 8, 15, 20$ respec-

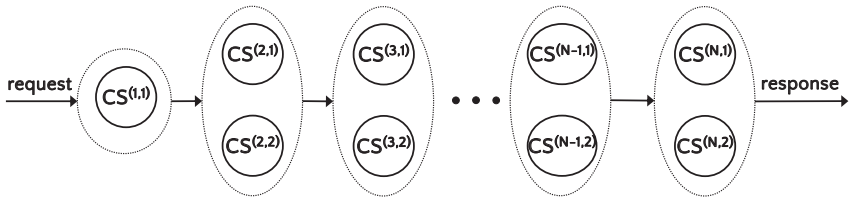


Figure 6.8: *Configuration for analyses on the impact of the service-chain length,* $M_1 = 1,\ M_i = 2,\ i = 2, \dots, N$.

tively with end-to-end probability $p_{e2e} = 90\%$ for the reference workflow. In these figures different gray-scale levels of the contour areas identify different benefit gain G. Each grey-scale area corresponds to a specific range of relative gain G as specified in the color bar. The arrows identify the (dashed) lines that separate different optimal static policies e.g. the number of faster alternatives used. We observe wedge shapes at the edges of the gray-scale contour areas. These wedges correspond to the switching curves where the static policy adds another faster alternative to the workflow, like the wedge shaped curves around the switching curve in section 6.5.1. Red lines identify the switching curves between different optimal static workflows. Arrows, left from the vertical axis, identify what static optimal paths are separated.

Similar to Section 6.5.1, more gain is found around the static optimal path switching curves. Switching curves have an expression similar to (6.18): for $k = 0, \ldots, N-1$,

$$K_V + K_R = \frac{c_{W_k} - c_{W_{k+1}}}{\overline{c}(p_{W_k} - p_{W_{k+1}})}. \tag{6.18}$$

p_{W_k} is the probability that a static workflow with k faster alternatives will generate a response within deadline δ_p. Observe that if the service-chain length N increases then the grey-scale contour areas become wider. This implies that longer workflows potentially have a higher benefit gain G when comparing dynamic and static workflows.

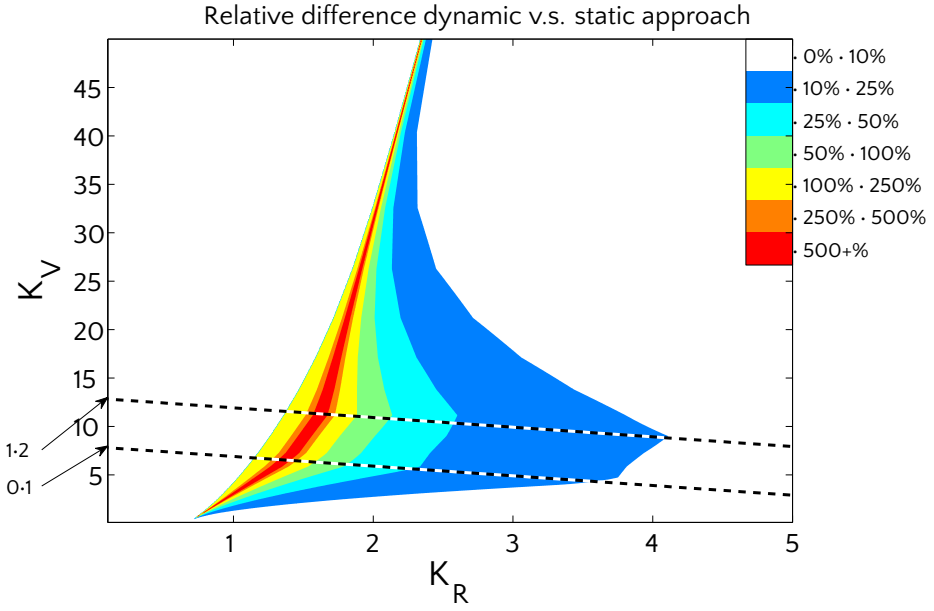


Figure 6.9: *Contour plot for the relative gain G with $N = 3$ and $p_{e2e} = 90\%$.*

6.5.4 Arbitrary alternatives

We consider a service workflow illustrated in Figure 6.14 for our experiments. This sequential workflow consists of $N = 4$ tasks. For each task i, there are M_i concrete service alternatives, where M_i could take one of the values $\{1, 2, 3, 4\}$, and $M_i \neq M_j$ whenever $i \neq j$. The notation (M_1, M_2, M_3, M_4) depicts the particular experimental setup, and represents one of the possible permutations of the set $\{1, 2, 3, 4\}$. Therefore, there are in total 24 different compositions (experimental setups).

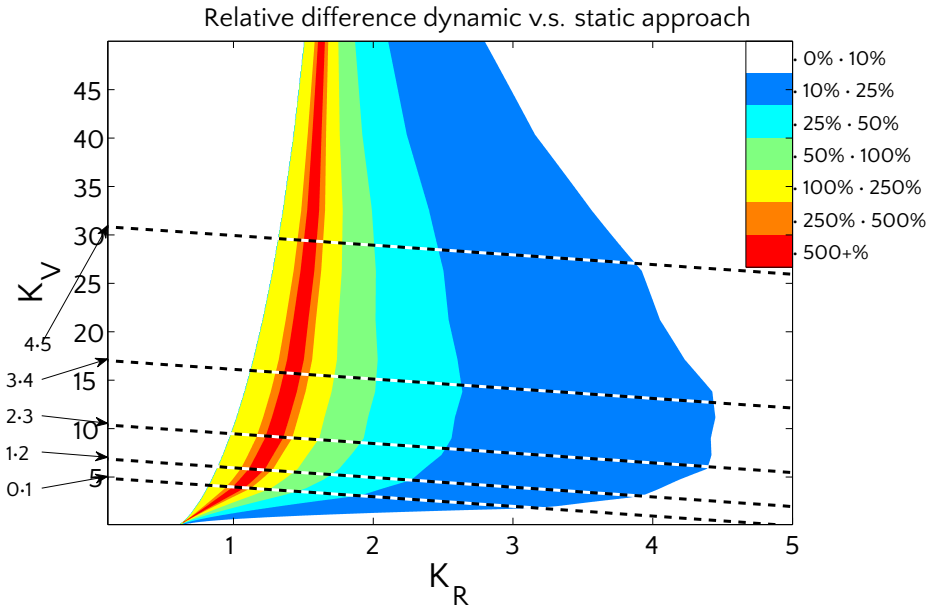


Figure 6.10: *Contour plot for relative gain G with* $N = 6$ *and* $p_{e2e} = 90\%$.

Let $\boldsymbol{W}_{SW} = [w_1, w_2, w_3, w_4]$ represent the service composition for the Static Workflow (SW) strategy, where $1 \leq w_j \leq M_j,\ j = 1, 2, 3, 4$. We want to compare the SW and DP strategies with fixed scaled cost parameter K_V and scaled reward parameter K_R as defined in (6.15). Let $W_{statopt}$ be the optimal static service composition. We choose to dimension the deadline δ_p such that the optimal static composition has at least a fraction of successful requests within the δ_p equal to p_{e2e}:

$$\mathbb{P}(D_{W_{statopt}} < \delta_p) \geq p_{e2e}. \tag{6.19}$$

Furthermore we tailor the reward and penalty parameters for optimal static composition "path" such that the expected reward R^*_{SW} for the static composition equals a predefined value. For the SW strategy R^*_{SW} is given by

$$R^*_{SW} = R \cdot p_{e2e} - V \cdot (1 - p_{e2e}) - c_{\boldsymbol{W}_{SW}}. \tag{6.20}$$

The expression above is similar to (6.5) with workflow cost $c_{\boldsymbol{W}_{SW}}$ as defined in (6.4). In Section 6.5.1 we observed that most relative gain in benefit can be expected where the expected benefit of static composition is small. Therefore, we use the SW strat-

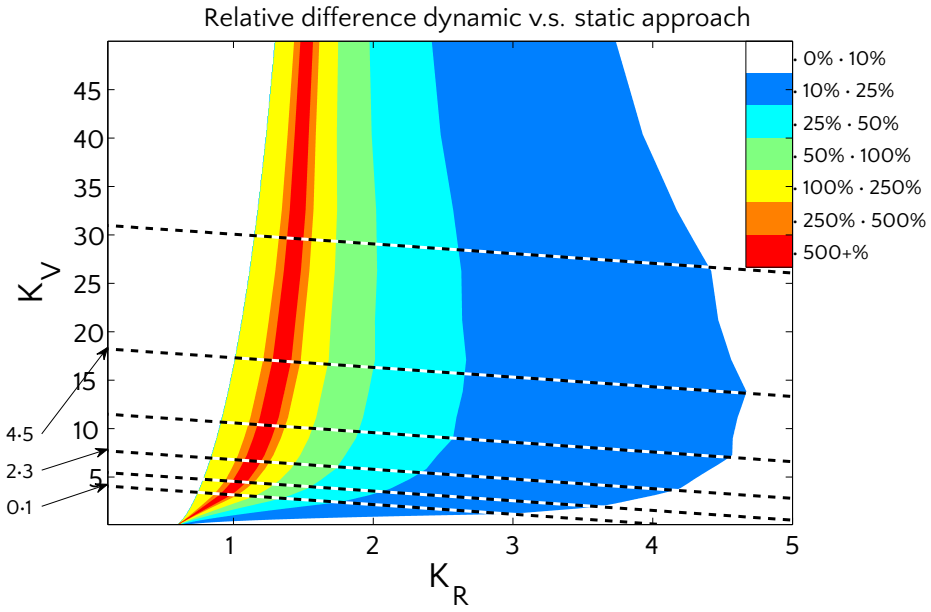


Figure 6.11: *Contour plot for relative gain G with* $N = 8$ *and* $p_{e2e} = 90\%$.

egy for the benchmarking and we set the value $R^*_{SW} = 0.01$. Taking into account (6.20) the values for parameters R and V satisfy the following equations:

$$R = \frac{R^*_{SW} + c_{\boldsymbol{w}_{SW}} + K_V \cdot (1 - p_{e2e}) \cdot \overline{c}}{p_{e2e}}, \qquad (6.21)$$
$$V = \frac{R^*_{SW} + c_{\boldsymbol{w}_{SW}} - K_R \cdot p_{e2e} \cdot \overline{c}}{p_{e2e} - 1}.$$

We have conducted simulations for two general scenarios: symmetric and asymmetric. These scenarios are defined based on the selection of cost parameters, as well as μ and σ^2 for a concrete service.

The goal of the symmetric scenario simulations is to illustrate the importance of the position of, and the number of, service alternatives within the workflow. The choice of the parameters for symmetric scenario is given in Table 6.6; here the parameters of the concrete service alternatives are the same for all tasks. When the number of alternatives for task is e.g. two, we always consider alternative 1 and alternative 2 in our experiments.

In the asymmetric scenario, which corresponds to parameter Table 6.7, the concrete services have different mean response times for tasks at different positions in the

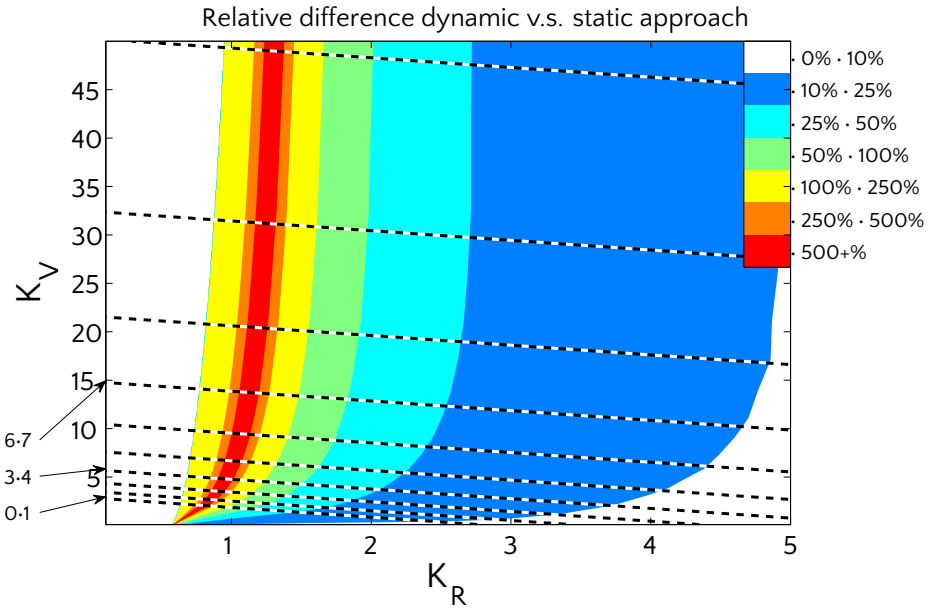


Figure 6.12: *Contour plot for relative gain G with $N = 15$ and $p_{e2e} = 90\%$.*

		Task $(i, \cdot)$		
		$\mathbf{i} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4
	$(\cdot, \mathbf{2})$	5	2.5	4
	$(\cdot, \mathbf{3})$	10	1.25	16
	$(\cdot, \mathbf{4})$	50	0.5	0.0009

Table 6.6: *Concrete service alternatives symmetric scenario*

.

service composition chain. For example at position 2, alternative $j = 1$ has cost $c^{(2,1)} = 5$, mean $\mu_{2,1} = 10$ and variance $\sigma^2_{2,1} = 16$, while alternative $j = 2$ is cheaper (, i.e. $c^{(2,2)} = 1$), has a lower mean $\mu_{2,2} = 9.5$ but higher variance, $\sigma^2_{2,2} = 64$. Furthermore, at position 3 an expensive service with zero variance is added. The goal of the asymmetric scenario is to illustrate the importance of the variance in addition to mean and cost for the service selection, which is usually neglected in state of the art service composition solutions.

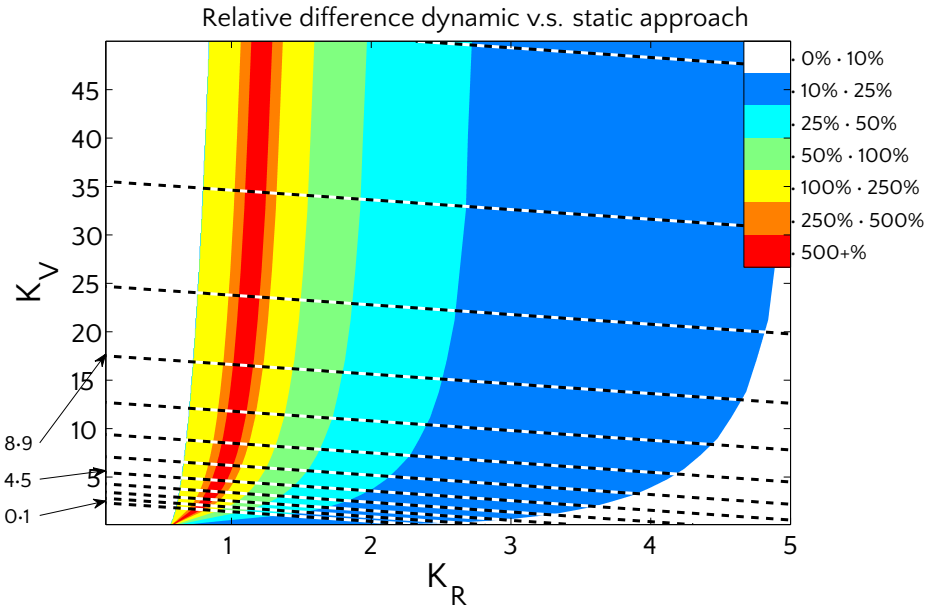


Figure 6.13: *Contour plot for relative gain G with* $N = 20$ *and* $p_{e2e} = 90\%$.

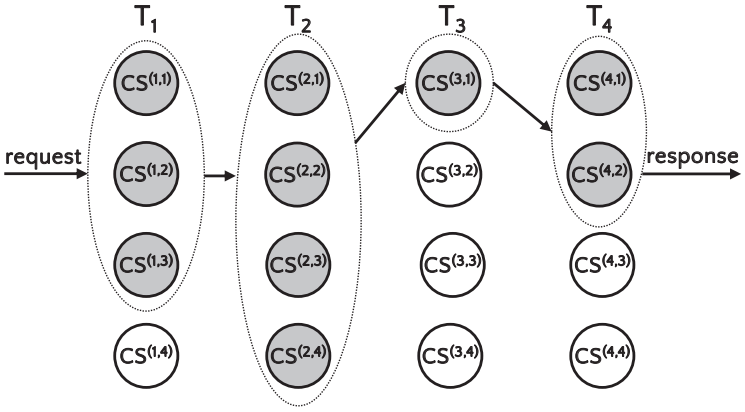


Figure 6.14: *Example workflow where number of alternative concrete services are represented by permutation* $(M_1, M_2, M_3, M_4) = (3, 4, 1, 2)$.

The calculated values for reward and penalty parameters (and given $p_{e2e} = 0.9$) are then used for the simulations of the DP strategy, for both symmetric and asymmetric scenarios.

6.5.5 Results

For all possible permutations of the number of concrete service alternatives (see Table 6.8) the expected benefit $\mathbb{E}[R]$ is calculated for both symmetric and asymmetric scenarios and DP and SW algorithms. The results are summarized in Figure 6.15 (symmetric scenario) and Figure 6.16 (asymmetric scenario). For both figures, the alternative count configurations are ordered on expected benefit for the DP algorithm.

In Figure 6.15 we observe that the DP solution achieves the lowest benefit for the permutation $(M_1, M_2, M_3, M_4) = (4, 3, 2, 1)$. In this case, there are many alternatives at the first position and no alternatives at the end of the workflow resulting in no possibility to recover from (possible) large service response time accumulated from the previous tasks in the workflow. Furthermore, the DP-based solution always picks the same alternative for the first service in the workflow, not taking any advantage of available alternatives for this service. On the other hand, the highest benefit for the DP solution is achieved for the permutation $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$. The largest number of alternatives are then available at the last service within the workflow, thus increasing the possibility to recover from (possible) large response times accumulated at the beginning of the workflow. From Figure 6.15 seven regions can be identified labeled by capital letters *A*, *B*, *C*, *D*, *E*, *F*, *G*. Each region is separated by line where the configuration change resulted in a significant increase in expected benefit. The changes that correspond to the most significant increase in benefit are listed in Table 6.10. In this table, labels M_3 and M_4 correspond to the number of alternatives available at the end of the workflow. We observe from Figure 6.15 that six configurations in regions *F* and *G* with the highest benefit are those where the

		Task $(i, \cdot)$					
		$(\mathbf{1}, \cdot)$			$(\mathbf{2}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4	5	10	16
	$(\cdot, \mathbf{2})$	5	2.5	4	1	9.5	64
	$(\cdot, \mathbf{3})$	10	1.25	16	10	1.5	0.25
	$(\cdot, \mathbf{4})$	50	0.5	0.0009	50	1	0.0025

		Task $(i, \cdot)$					
		$(\mathbf{3}, \cdot)$			$(\mathbf{4}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	0.5	0.04	1	2.5	1
	$(\cdot, \mathbf{2})$	5	0.4	0.04	5	2.45	2.25
	$(\cdot, \mathbf{3})$	10	0.3	0.0025	10	1	4
	$(\cdot, \mathbf{4})$	100	0.05	0	50	0.25	0.0004

Table 6.7: *Concrete service alternatives asymmetric scenario.*

		label											
		a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**
position	**1**	4	3	4	2	3	2	4	3	4	1	3	1
	2	3	4	2	4	2	3	3	4	1	4	1	3
	3	2	2	3	3	4	4	1	1	3	3	4	4
	4	1	1	1	1	1	1	2	2	2	2	2	2

		label											
		m	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**
position	**1**	4	2	4	1	2	1	3	2	3	1	2	1
	2	2	4	1	4	1	2	2	3	1	3	1	2
	3	1	1	2	2	4	4	1	1	2	2	3	3
	4	3	3	3	3	3	3	4	4	4	4	4	4

Table 6.8: *Indices of alternative configurations.*

number of alternatives for task 4 is the highest. Configurations in region *G* perform better than configurations *F* as the number of alternatives for task 3 is 3 in region *G*, and either 1 or 2 in region *F*. Additionally, the benefit "jump" between regions *E* and *F* is due to the number of alternatives (3 and 4, respectively) for task 4. The "jump" between regions *D* and *E* is caused by the fact that in region *D* for abstract service 3 only alternatives 1 and 2 are available while for region *E* 4 alternatives are available for task 3. The largest DP benefit improvement is when for the task $i = 4$, concrete service alternative 4 is considered (see Table 6.6). When this alternative with low mean and variance is considered, enough certainty to proceed with the workflow execution exists and the high price of concrete alternative 4 will be compensated by the increase in expected benefit. Another (smaller) benefit increase can be observed when concrete service alternative 3 becomes available for task at position 3. This can be explained by the fact that the 90th percentile for alternative 3 is still lower than 1 and 2 (see Table 6.9) despite its higher variance.

	Alternative			
	1	**2**	**3**	**4**
Percentile	7.61	4.81	2.74	0.54
μ	5	2.5	1.25	0.5
σ	2	2	4	0.03

Table 6.9: *Concrete service alternative 90th percentile.*

In Figure 6.16 the results are given for the asymmetric scenario. For the asymmetric scenario it is harder to observe any structure, because the different alternatives have different impact on the response time and the DP takes advantage of the properties of all available concrete service alternatives.

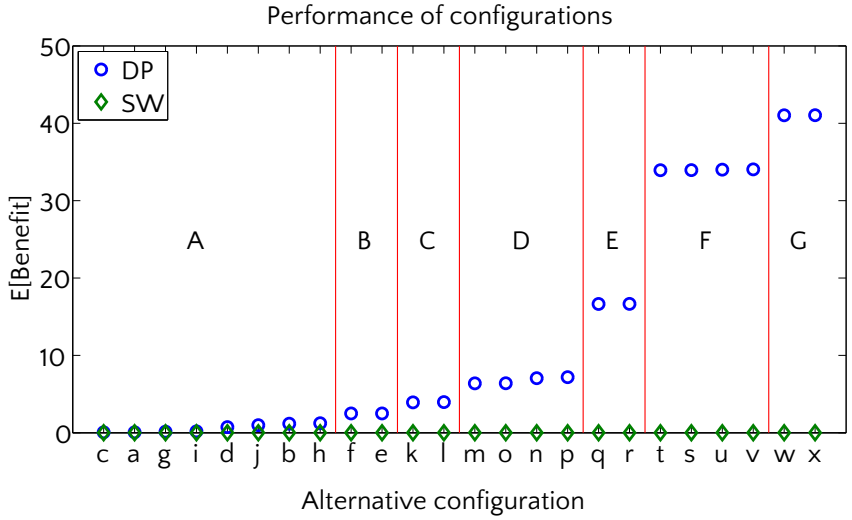


Figure 6.15: *Expected benefits per request in case of asymmetric scenario; comparison between static SW (static workflow) and DP (dynamic programming) for different configurations. See Table 6.10 for quick comparison of characteristics for regions D, E, F, G.*

		Region			
		D	**E**	**F**	**G**
Task *i* #	**M_3**	< 3	4	< 3	3
alternatives: M_i	**M_4**	3	3	4	4

Table 6.10: *Symmetric scenario regions from Figure 6.15.*

For alternative configurations $(M_1, M_2, M_3, M_4) = (4, 3, 2, 1)$ and $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$ the lowest expected and the highest benefit are achieved, respectively. The largest benefit increase is achieved when the near-zero variance service alternative is considered at position 4 and when the cheaper second alternative at position 2 is included despite its higher variance.

In Figure 6.17 we show the comparison of the rewards for the symmetric and asymmetric scenarios. The comparison is done when the same service configuration is used for both scenarios. The indices of service configuration are shown in Table 6.8.

The main conclusions and some "rules of a thumb" that could be drawn from the Figures 6.15 - 6.17 are as the following:

- Dynamic, on-the-fly service composition results in higher benefits for the CSP, compared to optimal "static" service composition (Figures 6.15 and 6.16). While the expected reward per request for the SW strategy is 0.01, for the

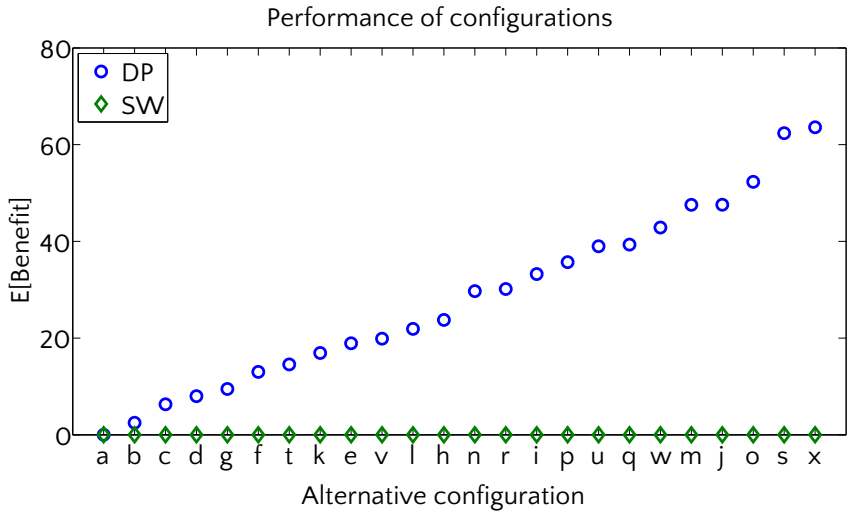


Figure 6.16: *Expected benefits per request in case of asymmetric scenario; comparison between static SW (static workflow) and DP (dynamic programming) for different configurations.*

symmetric and asymmetric DP scenarios, the expected rewards per request, depending of the configuration, may be greater than 40 or greater than 60, respectively.

- It is more beneficial to have a higher number of concrete service alternatives closer to the end of the sequential workflow (Figures 6.15 and 6.16).
- The variability of the response-times may have significant impact to the benefits achieved. When the end-to-end deadline is in jeopardy, it may be better to select a more expensive service with a smaller response-time variability (and smaller mean) than less expensive one with large response-time variability (Figure 6.16).
- In general it is better to have more response-time versatility with respect to mean and variance, (Figure 6.17). However, this needs to be investigated further.

6.6 Discussion

In this work, we have developed a model to maximize benefit for composite services by on-the-fly dynamic service selection. The selection decisions are based

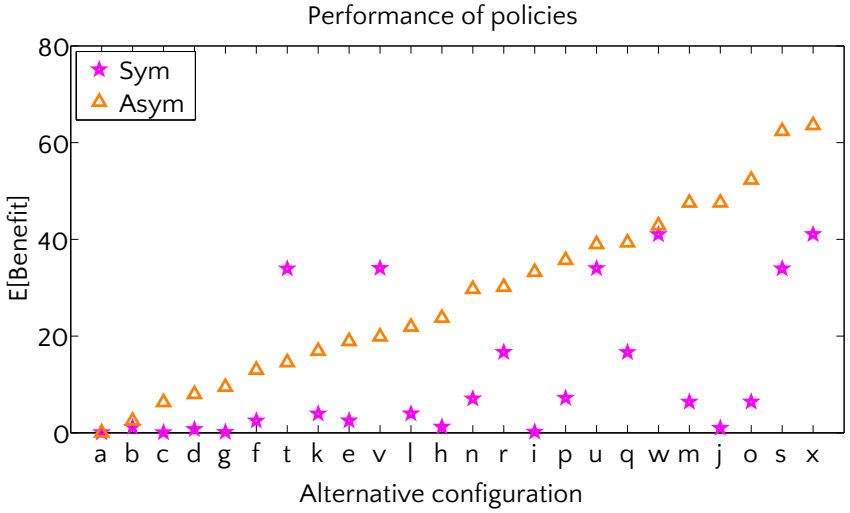


Figure 6.17: *Benefit comparison between symmetric and asymmetric scenarios.*

on observed response times, the response-time characteristics of the alternative, the end-to-end response-time objectives, and the reward and penalty parameters. The results not only indicate *that* there is enormous potential gain compared to other, non-dynamic approaches, but also show *how* one can realize such gain. We believe that this work is a significant step in realizing cost-efficient provisioning of complex composite services.

We end this chapter with a number of challenging areas for follow-up research.

First, in this chapter we have considered the case when SLAs are time invariant, i.e. once established, response-time SLOs, represented by respective PDFs, do not change (see also Remark 6.3.1). However, in practice the response-time distributions may be subject to change over time (e.g., due temporary failure or overload of a concrete service). Moreover, the empirical PDFs are typically determined based on response-time measurements over a finite time horizon, and hence will change over time. In this context, an important next step is to devise and implement models and methods to support *closed-loop control*, where the lookup table is re-calculated in an optimal way (e.g., with respect to the frequency of updates). Note that the results presented in this chapter form an excellent basis for extension towards closed-loop controlled system.

Second, the DP solution may tend to slow down when number of services is extremely large. Therefore, to provide good service quality for complex composite services composed of many services, there is a need for the development of fast yet efficient heuristic solutions, which opens up a challenging area for further research.

Third, in this model a specific reward-and-penalty cost structure was assumed, where the CSP pays an amount of money to the ISP for the execution of a single request, and gets a reward for execution of a single request within penalty deadline but pays penalty to the end customer when the agreed end -to-end deadline is not met. In practice, many other cost structures (e.g., discount structures) are conceivable. Extension of the results to include other cost structures is an interesting area for follow-up research.

Finally, another interesting and practically useful extension of the model is to include the possibility of re-attempts when the response-time of a given individual service exceeds some threshold value. Such reattempts may be particularly beneficial when the response-time distribution has a decreasing hazard rate of failure. Investigation of the potential for cost reduction and the cost-benefit trade-off of reattempts is another promising venue for further research.

Appendix 6.A Workflow aggregation example

We illustrate some basic aggregation rules using an example workflow (represented by Figure 6.18) and show how it can be mapped into the (relatively simple) sequential workflow. The proposed aggregation could be done *as long as there is no data dependence among the services*. The aggregation rules are given for the case when probabilistic (i.e. stochastic) QoS models are used for different web service QoS parameters, e.g. response-time. The calculation of the composite service QoS has been analysed in many papers (e.g., [96, 29, 66, 63, 118]). However, none of these papers considered stochastic QoS models, except [96], in which the authors calculate the composite service QoS parameters using Monte-Carlo simulations.

The workflow in Figure 6.18 consists of four elementary but frequently used workflow composition patterns, namely sequential, flow, switch and loop. There are two alternatives for tasks 1, 2 and 5, three alternatives for task 3, and tasks 4 and 6 have one alternative. The flow pattern represents (part of a) workflow in which all tasks are executed in parallel, and the response is not available till all services finish their execution (tasks 2 and 3 at Figure 6.18). The switch statement represents workflow that executes one of the tasks with given probabilities. Referring back to Figure 6.18 we identify the switch pattern for tasks 4 and 5, which are executed with probabilities p_4 and p_5, respectively, where $p_4 + p_5 = 1$. In general, the loop control statement consists of K consecutive invocations of the single task (service 6 in Figure 6.18). Each service within the workflow is represented by an probabilistic response-time SLO, i.e. the response-time probability-density function (PDF) and/or response-time cumulative distribution function (CDF). For concrete service j of task i within the given workflow, the PDF and CDF are $f^{(i,j)}(t)$ and $F^{(i,j)}(t)$, respectively. The execution cost in this case is $c^{(i,j)}$, and response-time random

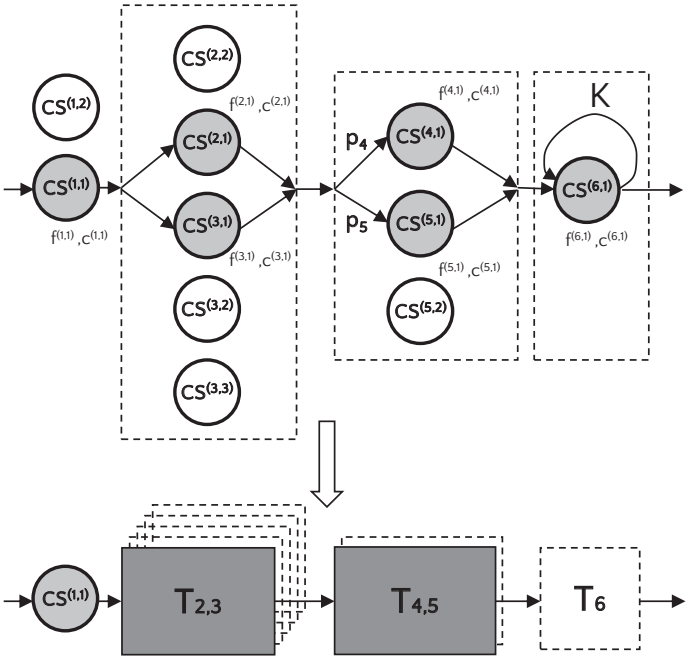


Figure 6.18: *Example workflow reduced and aggregated into the sequential workflow.*

variable is $D^{(i,j)}$. The aggregation rules for concrete services depicted in Figure 6.18 are as follows:

- The resulting response-time PDF for the flow pattern is expressed by the term $f^{T_{2,3}}(t) = F^{(2,1)}(t) \cdot f^{(3,1)}(t) + f^{(2,1)}(t) \cdot F^{(3,1)}(t)$. The execution cost is given as $c^{T_{2,3}} = c^{(2,1)} + c^{(3,1)}$.
- The resulting response-time PDF for the switch pattern (services 4 and 5) is given as $f^{T_{4,5}}(t) = p_4 \cdot f^{(4,1)}(t) + p_5 \cdot f^{(5,1)}(t)$. The execution cost is $c^{T_{4,5}} = p_4 \cdot c^{(4,1)} + p_5 \cdot c^{(5,1)}$.
- The resulting response-time PDF of the loop pattern is expressed as the K-fold convolution of the PDF $f^{(6,1)}(t)$, i.e. $f^{T_6}(t) = f^{(6,1)}(t) * f^{(6,1)}(t) * \cdots * f^{(6,1)}(t) = \left[f^{(6,1)}(t)\right]^K$, where $*$ represents convolution operator. The CDF $F^{T_6}(t)$ is calculated similarly, and the execution cost is $K \cdot c^{(6,1)}$.

For the case where aggregation of the tasks with multiple alternatives the aggregation should take place for each combination of the alternatives for considered tasks. Therefore, aggregation of tasks 2 and 3, $T_{2,3}$ has 6 alternatives, $T_{4,5}$ has 2 alternatives, and so on. The response-time PDFs for the aggregation of more complex workflow

patterns could be efficiently numerically calculated using the methods described in [34].

Autonomous Runtime QoS Control for Composite Services in SOA

7

In this chapter, we propose a runtime closed loop control mechanism that dynamically optimizes service composition in real time by learning and adapting to changes in third party service response time behaviors.

Negotiating multiple SLAs in itself is not sufficient to guarantee end-to-end QoS levels as SLAs in practice often give probabilistic QoS guarantees and SLA violations can still occur. Moreover probabilistic QoS guarantees do not necessarily capture time-dependent behavior, (e.g., short term service degradations). Therefore, the negotiation of SLAs needs to be supplemented with *run-time QoS-control* capabilities that give providers of composite services the capability to properly respond to short-term QoS degradations (real-time composite service adaptation). Motivated by this we developed an approach that captures (temporarily) QoS degradations by tracking the behavior of third party services.

In our approach response-time realizations are used for learning an updating the response-time distributions. The currently known response-time distribution is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if an significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. The main goal of this run-time service selection and composition is benefit maximization for the composite service provider and ability to adapt to changes in response-time behavior of third party services. To demonstrate the usefulness of the mechanism, we have implemented our approach in a simulation environment. Moreover, we evaluate the influence of the control parameter settings on the effectiveness of our control mechanism.

7.1 Background

By tracking response times the actual response-time behavior can be captured in empirical distributions. In [122] we apply dynamic programming (DP) and we derive a service-selection policy based on response-time realizations. With this approach we assume that the response-time distributions are known or derived from histori-

[7] This chapter is based on [21], [120] and [122].

cal data. This results in a lookup-table which determines what third party alternative should be used based on actual response-time realizations.

We extend this work such that we can learn an exploit response-time distributions on the fly. Reinforcement-learning techniques are known to be able to do this but our model has a special structure that complicates the use of the classical TD learning approaches. The solution of our DP formulation searches the stochastic shortest path in a stochastic activity network [34]. Typically RL techniques solve complex learning and optimization problems by using a simulator. This involves a Q value that assigns utility to state-action combinations. Most algorithms run off-line as a simulator is used for optimization. RL has also been widely used in on-line applications. In such applications, information becomes available gradually with time. Most RL approaches are based on environments that do not vary over time. We refer to [53] for a good survey on reinforcement learning techniques.

The dynamic program (DP) in our solution has a special structure, such that the solution of a smaller sub-problem can be used to solve our problem at a more complex level. This DP can be characterized as a hierarchical DP [53, 8]. Therefore classical RL is not suitable and hierarchical RL has to be applied [8]. Also changes in response-time behavior are likely to occur which complicates the problem even more. Both the problem structure and volatility are challenging areas of research in RL.

In our approach we tackle both the hierarchical structure, and time varying behavior challenges. To this end we are using empirical distributions and updating the lookup table if significant changes occur. As we are considering a sequence of tasks, the number of possible response time realizations combinations explodes. By discretizing the empirical distribution over fixed intervals we overcome this issue. Furthermore this enables the approach where the empirical distribution is updated using a smoothing approach. An advantage of this is that no bookkeeping of previous response-time values is necessary as all samples are already represented in the smoothed distribution.

The remainder of this chapter is as follows. We start in section 7.3 with the introduction of our closed loop control learning approach. Next in Section 7.2 we introduce the workflow model that needs to be optimized. In Section 7.4 we introduce the tools for handling empirical distributions and change point detection. Using the Empirical distributions we define comprising a dynamic program that optimizes expected benefit. Experiments on our approach are performed in Section 7.5. Results are presented and discussed in Section 7.6. Finally we conclude in Section 7.7.

7.2 Model

We consider a composite service that comprises a sequential workflow consisting of N tasks identified by $T_1, \ldots, T_N$. The tasks are executed one-by-one in the sense that each consecutive task has to wait for the previous task to finish. Our solution is applicable to any workflow that could be aggregated and mapped into a sequential one. Basic rules for aggregation of non-sequential workflows into sequential workflows have been illustrated in, e.g. [122, 119, 34]. However, the aggregation leads to coarser control, since decisions could not be taken for a single service within the aggregated workflow, but rather for the aggregated workflow patterns themselves.

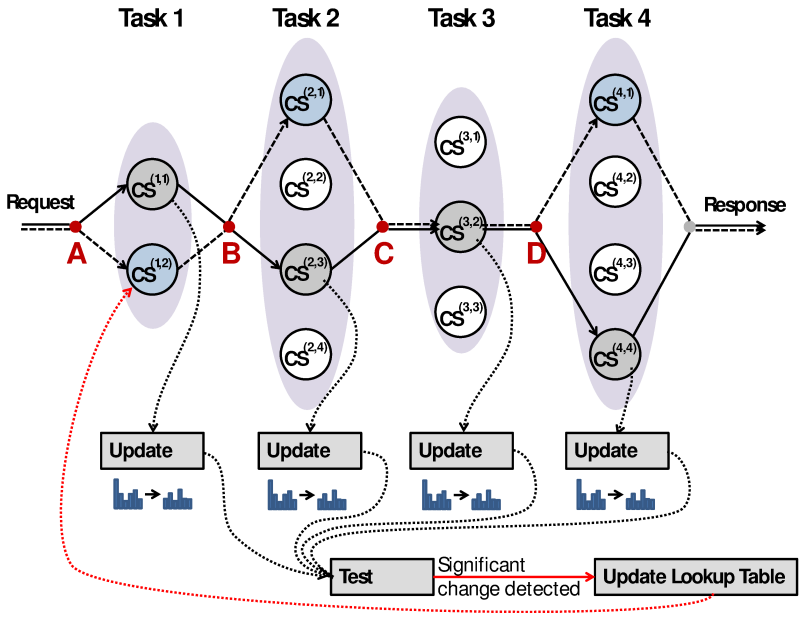


Figure 7.1: *Orchestrated composite web service depicted by a sequential workflow. Dynamic run-time service composition is based on a lookup-table. Decisions are taken at points A-D. For every used concrete service (CS) the response-time distribution is updated with the new realization. In this example a significant change is detected. As a result for the next request concrete service 2 is selected at Task 1.*

The workflow is based on an unambiguous functionality description of a service (``abstract service''), and several functionally identical alternatives (``concrete services'') may exist that match such a description [93]. Each task has an abstract

service description or interface which can be implemented by external service providers.

The workflow in Figure 7.1 consists of four tasks, and each task maps to a number of concrete services (alternatives), which are deployed by (independent) third-party service providers. For each task T_i there are M_i concrete service providers $CS^{(i,1)}, \ldots, CS^{(i,M_i)}$ available that implement the functionality corresponding to task T_i. For each request processed by $CS^{(i,j)}$ cost $c^{(i,j)}$ has to be paid. Furthermore there is an end-to-end response-time deadline δ_p. If a request is processed within δ_p a reward of R is received. However, for all requests that are not processed within δ_p a penalty V had to be paid. After the execution of a single task within the workflow, the orchestrator decides on the next concrete service to be executed, and composite service provider pays to the third party provider per single invocation. The decision points for given tasks are illustrated at Figure 7.1 by A, B, C and D. The decision taken is based on (1) execution costs, and (2) the remaining time to meet the end-to-end deadline. The response time of each concrete service provider $CS^{(i,j)}$ is represented by the random variable $D^{(i,j)}$. After each decision the observed response time is used for updating the response time distribution information of the selected service. Upon each lookup-table update the corresponding distribution information is stored as reference distribution. After each response the reference distribution is compared against the current up-to date response time distribution information.

7.3 Closed loop control

In this section we explain our closed loop approach. The main goal of this approach is benefit maximization for the composite service provider, and ability to adapt to changes in response-time behavior of third party services. We realize this by monitoring/tracking the observed response-time realizations. In the introduction we explained that the response-time based selection comprises a lookup-table that is calculated using DP (see Section 7.4.1). The DP needs information about response-time distributions and costs in order to determine lookup-table. This lookup-table corresponds to a strategy that optimizes expected benefit. In our approach, observed response-time realizations are used for learning an updating empirical response-time distributions. The currently known response-time distribution is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if an significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. We illustrate our approach using Figure 7.2.

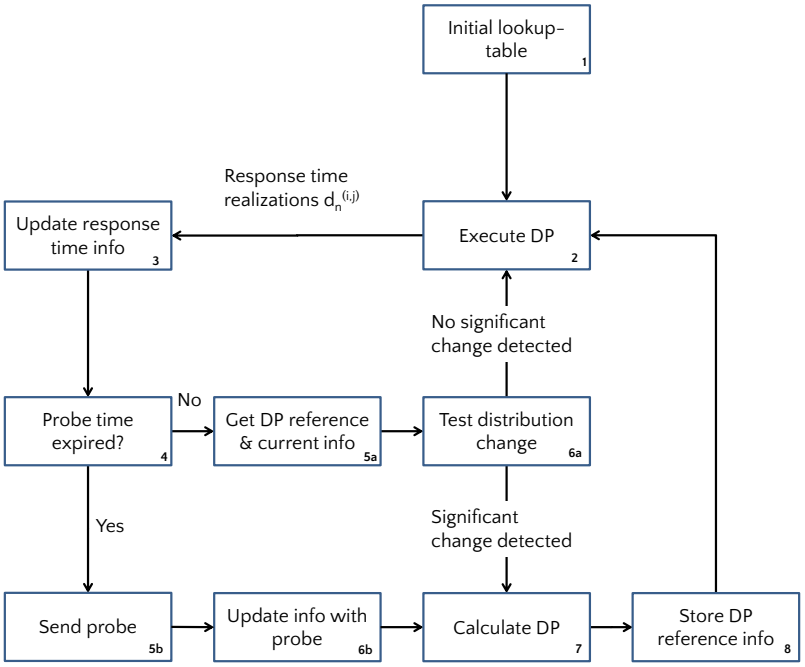


Figure 7.2: *Closed loop control approach.*

The execution starts with an initial lookup table at step (1). This could be derived from initial measurements on the system. After each execution of a request in step (2) the empirical distribution is updated at step (3). A DP based lookup-table could leave out unattractive concrete service providers. In that case we do not receive any information about these providers. These could become attractive if the response-time behavior changes. Therefore in step (4), if a provider is not visited for a certain time, a probe request will be sent at step (5b) and the corresponding empirical distribution will be updated at step (6a). After each calculation of the lookup-table, the current set of empirical distributions will be stored. These are the empirical distributions that were used in the lookup-table calculation and form a reference response-time distribution. Calculating the lookup-table for every new sample is expensive and undesired. Therefore we propose a strategy where the lookup-table will be updated if a significant change in one of the services is detected. For this purpose the reference distribution is used for detection of response-time distribution changes. In step (5a) and step (6a) the reference distribution and current distribution are retrieved and a statistical test is applied for detecting change in the response-time distribution. If no change is detected then the lookup-table remains unchanged. Otherwise the lookup-table is updated using the DP. After a probe update in step (5b) and step

(6b) we immediately proceed to updating the lookup-table as probes are sent less frequently. In step (7) and step (8) the lookup-table is updated with the current empirical distributions and these distributions are stored as new reference distribution. By using empirical distributions we are directly able to learn and adapt to (temporarily) changes in behavior of third party services.

Using a lookup-table based on empirical distributions could result in that certain alternatives are never invoked. When other alternatives break down this alternative could become attractive. In order to deal with this issue we use probes. A probe is a dummy request that will provide new information about the response time for that alternative. As we only receive updates from alternatives which are selected by the dynamic program, we have to keep track of how long ago a certain alternative has been used. For this purpose to each concrete service provider a probe timer $U^{(i,j)}$ is assigned with corresponding probe time-out $t_p^{(i,j)}$. If a provider is not visited in $t_p^{(i,j)}$ requests ($U^{(i,j)} > t_p^{(i,j)}$) then the probe timer has expired and a probe will be collected incurring probe cost $c_p^{(k,j)}$. If for example, in Figure 7.1, the second alternative of the third task has not been used in the last ten requests, the probe timer for alternative two has value $U^{(3,2)} = 10$. After a probe we immediately update the corresponding distribution. No test is applied here as probes are collected less frequent compared to processed requests. Probe cost introduces a typical trade-off between benefit due to up-to date information and cost on acquiring information. In Section 7.5 we explore the probe frequency-cost trade-off.

7.4 Algorithms

In this section we elaborate on the algorithms and detection mechanisms that are used in the closed loop control approach. These include dynamic programming (DP) in Section 7.4.1, empirical distribution discretization in Section 7.4.2, sliding window smoothing in Section 7.4.3 and exponential smoothing in Section 7.4.4. In Sections 7.4.3, and 7.4.4 we also cover change point detection using the Kolmogorov-Smirnov statistical test.

7.4.1 Determining the lookup-table

We now modify the dynamic program in Section 6.4 that calculates the optimal strategy given the current empirical distributions. Therefore we need distretized empirical distributions. Let h be the discretization step size. Let T^* be the end to end deadline: $T^* = \left\lceil \frac{\delta_p}{h} \right\rceil$. Furthermore we define $q_k^{(i,j)}$ as the discretized empirical distribution of concrete service alternative j at task j at $t = hk$. Using the discretization

approach of [34] we discretized the empirical distributions as follows for $i = 1, \dots, N$, $j = 1, \dots, M_i$, $k = 0, \dots, T^*$:

$$q_k^{(i,j)} = \begin{cases} \sum_{t=1}^{W} \mathbb{1}\left\{h[k-0.5] < d_{n-t}^{(i,j)} \leq h[k+0.5]\right\} & \text{if } k < T^*, \\ \sum_{t=1}^{W} \mathbb{1}\left\{d_{n-t}^{(i,j)} > h[k-0.5]\right\} & \text{otherwise.} \end{cases} \tag{7.1}$$

Here is $d_t^{(i,j)}$ the t-th response-time realization for service alternative j at task i, and $\mathbb{1}_{\{A\}}$ is the indicator function over A which is 1 if A is true and 0 otherwise. More details about determining and using the discretized empirical distribution can be found in Section 7.4.2. Using the discretized empirical distributions backward recursion formulae can be formulated. We start with the terminal reward function for $b = 0, \dots, T^*$:

$$P_b^{(N+1,*)} = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{7.2a}$$

Using this function we iterate backwards using the following equations for $i = 1, \dots, N$, $j = 1, \dots, M_i$, $b = 0, \dots, T^*$:

$$P_b^{(i,*)} = \max_j \left\{ -c^{(i,j)} + R_b^{(i,j)} + V_b^{(i,j)} \right\}, \tag{7.2b}$$

$$R_b^{(i,j)} = \sum_{k=0}^{b} q_k^{(i,j)} P_{k-b}^{(i+1,*)} \quad \text{and} \tag{7.2c}$$

$$V_b^{(i,j)} = \sum_{k=b+1}^{T^*} q_k^{(i,j)} P_0^{(i+1,*)}. \tag{7.2d}$$

Here, the term $P_b^{(i,*)}$ represents the expected benefit per request given time budget b at task i under the optimal dynamic programming decision strategy. The term $R_b^{(i,j)}$ represents the expected reward, when concrete service j (assigned to task i) is executed for the given time budget value b. Finally the term $V_b^{(i,j)}$ represents the expected penalty for exceeding the overall deadline at task i while executing concrete service j for the given time budget value b. The expected reward and penalty functions take into account the impact of future decisions as represented by terms relating to $P_b^{(i+1,*)}$ in (7.2c) and (7.2d).

While applying formulae (7.2b)-(7.2d), the corresponding decisions (actions) $A_k^{(*,i)}$ can be obtained by storing the maximum arguments for $i = 1, \dots, N, \quad j = 1, \dots, M_i, \quad k = 0, \dots, T^*$:

$$A_k^{(i,*)} = \underset{j}{\operatorname{argmax}} \Big\{ -c^{(i,j)} + R_k^{(i,j)} + V_k^{(i,j)} \Big\}.$$

7.4.2 Empirical distribution

In 7.4.1 we introduced an approach for discretizing empirical distributions such that these become suitable for using in DP. This is derived from the discretization approach in [34] for $i = 1, \dots, N, \quad j = 1, \dots, M_i, \quad t = 0, \dots, T^*$:

$$\begin{aligned} q_k^{(i,j)} &= \begin{cases} \mathbb{P}\big(\hat{D}^{(i,j)} \le h[k+0.5]\big) - \mathbb{P}\big(\hat{D}^{(i,j)} \le h[k-0.5]\big), & k < T^*, \\ \mathbb{P}\big(\hat{D}^{(i,j)} > h[k-0.5]\big), & k = T^*, \end{cases} \\ &= \begin{cases} \sum_{t=1}^{W} \mathbb{1}\big\{h[k-0.5] < d_{n-t}^{(i,j)} \le h[k+0.5]\big\}, & k < T^*, \\ \sum_{t=1}^{W} \mathbb{1}\big\{h[k-0.5] < d_{n-t}^{(i,j)}\big\}, & k = T^*. \end{cases} \end{aligned} \tag{7.3}$$

In Equation (7.3), $\hat{D}^{(i,j)}$ is the empirical response-time process for concrete alternative j at task i, and
$d_n^{(i,j)}$ is the response time of the nth sample for concrete alternative j at task i.

Consider the discretized distribution $q_k^{(i,j)}$. Actually $q_k^{(i,j)}$ is a histogram where kth bin is bounded by
$\big[\, h(k-0.5); h(k+0.5) \,\big)$ and where the sum of the frequencies is normalized to 1. There is a tradeoff in choosing the bin size. When h has a small value the histogram will consist of many bins. In the case that a few large bins are used, too much information about sample location is lost. A usual method for constructing an empirical distribution histogram $\tilde{q}^{(i,j)_n}$ from samples is as follows: Let n be the number of samples that are already included in the histogram. Let $d_{n+1}^{(i,j)}$ be a new sample from the distribution we want to estimate. Then the histogram is updated as follows:

$$\tilde{q}_{n+1}^{(i,j)} = \frac{n}{n+1} \left[\tilde{q}_n^{(i,j)} + \mathbb{1}_{\big\{(k-0.5)h \le d_{n+1}^{(i,j)} < (k+0.5)h\big\}} \right], \tag{7.4}$$
$k = 0 \dots, K.$

note that this corresponds to the definition of the empirical distribution defined in (7.3). Define:

$$\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{[x_i \leq t]\}} \tag{7.5}$$

the empirical distribution over t based on samples x_i. The central limit theorem states that *pointwise*, $\hat{F}_n(t)$ has asymptotically a normal distribution (see [102, p265] for more details). Essentially $\mathbb{1}_{\{x_i \leq t\}}$ has a Bernoulli distribution with success probability $F(x_i \leq t)$.

$$\sqrt{n}\big(\hat{F}_n(t) - F(t)\big) \xrightarrow{d} \mathcal{N}\Big(0, F(t)\big(1 - F(t)\big)\Big). \tag{7.6}$$

Using the empirical distribution over all samples is a suitable approach when the distribution does not change over time. However, note that new samples will have a decreasing impact on the empirical distribution. If the distribution changes over time this is an undesired property.

7.4.3 Empirical distribution based on sliding window

A natural approach for tracking changes is that we define a sliding window of W samples. Let $\mathcal{X}_n = \{d_{n-n+1}, d_{n-W+2}, \dots, x_n\}$ be the current set of samples in the sliding window after inserting the nth sample d_n. These sets are used to determine the empirical distribution that serves as an input for the dynamic program. A disadvantage of this method is that the samples that are included in the sliding window need to be stored for the sliding window. The empirical distribution of the nth sample in the sliding window approach is defined by:

$$\hat{F}_n(t) = \frac{1}{W}\sum_{i=0}^{W-1} \mathbb{1}_{\{d_{n-i} \leq t\}}. \tag{7.7}$$

If we take a $\mathcal{X}_n$ and $\mathcal{X}_m$, $m > n$ we could apply statistical tests (e.g. the two sample Kolmogorov Smirnov test) in order to determine if a significant change occurred in the distribution. Let $D_{n,n'}$ be the Kolmogorov–Smirnov (KS) statistic on two empirical distributions with n and n' observations respectively. Let $\hat{F}(x)$ and $\hat{F}'(x)$ be the corresponding empirical distribution functions. Then the KS statistic is defined as:

$$D_{n,n'} := \sup_x |\hat{F}(x) - \hat{F}'(x)| \quad \text{and} \quad \sqrt{\frac{nn'}{n+n'}} D_{n,n'} \tag{7.8}$$

has a Kolmogorov distribution.

7.4.4 Exponentially smoothed empirical distribution

To prevent overhead of sliding window bookkeeping, we apply a smoothing approach. In this approach new samples can be included using the following weighting scheme:

$$\tilde{p}_{n+1}^{(i,j)} = \kappa \tilde{p}_n^{(i,j)} + (1-\kappa)\mathbb{1}_{\left\{(k-0.5)h \le d_{n+1}^{(i,j)} < (k+0.5)h\right\}}. \tag{7.9}$$

This is similar to the exponentially weighted moving average estimate in [32]. Here $0 < \kappa < 1$ is a smoothing factor. When κ is close to one new samples have a relatively small impact and it will take a long time before the empirical distribution converges to a new distribution. If κ is close to zero new samples have a big impact resulting in an empirical distribution that quickly follows changes in the real sample distribution but with more noise in the histogram.

Modified Kolmogorov-Smirnov test As we use smoothed empirical distributions, a statistical test on smoothed distributions is required. To this end we modified the two sample Kolmogorov-Smirnov test, suitable for comparing two smoothed empirical distribution functions $q_k^{(n)}$ and $q_k^{(m)}$. The Kolmogorov Smirnov test is based on the result that the empirical distribution as defined in (7.7) converges pointwise to a normal distribution then the number of samples goes to infinity. The original Kolmogorov Smirnov test relates this to the Brownian bridge on which the test statistic is based [102]. We want to apply a similar central limit theorem result for the smoothed process such that we can apply statistics like the Kolmororov Smirnov statistic. We try to relate κ to a virtual window size W such that the variance of any point in the smoothed empirical distribution corresponds to the variance of the original sliding-window empirical distribution consisting of W i.i.d. samples.

Let $q_k^{(n)}$, $q_k^{(m)}$ be empirical distributions, discretized according to (7.3), let X_i be i.i.d. random variables, and let Y_i be the smoothing process on the variables. For exponential smoothing (ES) the variance results from geometric terms:

$$\begin{aligned}\mathrm{Var}(Y_i) &= \kappa^2 \mathrm{Var}(Y_{i-1}) + (1-\kappa)^2 \mathrm{Var}(X_i) \\ &= (1-\kappa)^2 \sum_{k=0}^{\infty} \kappa^{2i} \mathrm{Var}(X_{i-k}) = \frac{1-\kappa}{1+\kappa} \mathrm{Var}(X_i).\end{aligned}$$

Thus for for given κ the virtual window size is $W = \frac{1+\kappa}{1-\kappa}$. Let K_α is the critical value of the Kolmogorov distribution corresponding to significance level α. We replace n and n' in Equation (7.8) with the virtual window size W:

$$\sqrt{\frac{1+\kappa}{2(1-\kappa)}} D_\kappa > K_\alpha. \tag{7.10}$$

The statistic is obtained from smoothed, discretized empirical distributions $q_k^{(n)}$, and $q_k^{(m)}$ as follows:

$$D_\kappa := \sup_k |q_k^{(n)} - q_k^{(m)}|. \tag{7.11}$$

7.5 Experimental setup

To test the closed-loop approach defined in Section 7.3 we define an experimental workflow that can be used for investigating the impact of the closed-loop control approach parameters. We emphasize that this experimental set-up is tailored for evaluating responsiveness of our closed-loop approach with respect to response-time distributions and does not limit us in tracking other systems with different response-time models.

Figure 7.3 represents a work flow consisting of four tasks: For each task four concrete

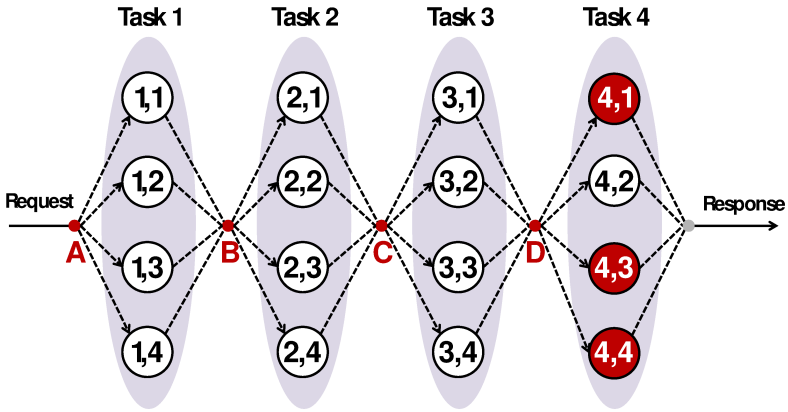


Figure 7.3: *Default experimental work flow. For task T_4 alternatives 1, 3 and 4 are slow. The optimal strategy should detect this and use alternative 2.*

service alternatives are available. At each alternative j for task T_i the response-time distribution is summarized in terms of mean $\mu^{(i,j)}$ and variance $\left(\sigma^{(i,j)}\right)^2$.

7.5.1 Experimental model

We model the burstiness of response-time behaviors using the classical Gilbert-Elliott [51, 45] Discrete Time Markov Chain (DTMC) model. Each concrete service is modeled with its own DTMC with underlying state which can be either fast or slow. For sake of readability we omit the superscript indexing for all variables in the experimental model definition ($\square^{(i,j)} \rightarrow \square$), but we emphasize that *each concrete service has its own set of parameters*. The DTMC enables us to capture the rate of change between fast and slow response-time behavior. Essentially the DTMC model results in taking a mixture of distributions represented by the fast and slow states. The DTMC is defined by the following transition matrix corresponding to state the vector $\begin{bmatrix} fast & slow \end{bmatrix}^T$ and corresponding state probabilities $\begin{bmatrix} p_{fast} & p_{slow} \end{bmatrix}^T$:

$$P = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}. \tag{7.12}$$

The transition probabilities α and β are parametrized by factors a, b, t_{cycle}, and k. In the parametrization $0 < a < 1$ is the scaling factor for the fast state mean μ_{fast}, $b > 1$ is the scaling factor for the slow state mean μ_{slow}. We define the expected cycle time $t_{cycle} > 0$ as the expected time it takes to leave the current state and return to its original state e.g. fast→slow→fast or slow→fast→slow. Furthermore the variances of the fast and slow states are related by scaling factor $k > 0$ where $k > 1$ if $\sigma^2_{fast} > \sigma^2_{slow}$ and $k < 1$ if $\sigma^2_{fast} < \sigma^2_{slow}$. Both high and low states are represented by log-normal distributions with means μ_{slow}, μ_{fast} and variances σ^2_{slow}, σ^2_{fast}. A motivation that supports this burstiness model is that random variables, holding times of various message types were found to be log-normal mixtures (conversation time in land and mobile telephone networks, voice mail message length, facsimile transmission time) [17] The log-normal distributions are related to the parametrization as follows:

$$\begin{aligned} \mu_{fast} &= a\mu, \qquad \mu_{slow} = b\mu, & 0 < a < 1, \quad 1 < b, \\ \sigma^2_{fast} &= v\sigma^2, \qquad \sigma^2_{slow} = kv\sigma^2, & v > 0, \quad k > 0. \end{aligned}$$

In order to obtain the desired mean μ and variance σ^2 we need:

$$\mu = p_{slow}\mu_{slow} + p_{fast}\mu_{fast}, \tag{7.13}$$

$$\sigma^2 + \mu^2 = p_{fast}(\sigma^2_{fast} + \mu^2_{fast}) + p_{slow}(\sigma^2_{slow} + \mu^2_{slow}). \tag{7.14}$$

From Equations (7.13) to (7.14) and the properties $\alpha p_{fast} = \beta p_{slow}$, and $\frac{1}{\alpha} + \frac{1}{\beta} = t_{cycle}$ we obtain the following parameter values in terms of a, b, t_{cycle}, and k:

$$\alpha = \frac{p_{fast}}{t_{cycle}}, \quad \beta = \frac{p_{slow}}{t_{cycle}}, \qquad p_{fast} = \frac{1-b}{a-b}, \qquad p_{slow} = 1 - p_{fast},$$
$$v = \frac{\sigma^2 + \mu^2(1 - p_{fast}a^2 - p_{slow}b^2)}{\sigma^2(p_{fast} + kp_{slow})}.$$

7.5.2 Experimental set-up

The parameters $\mu^{(i,j)}$, $\sigma^{(i,j)}$, $c^{(i,j)}$ of our experiments (Figure 7.3) are summarized in Table 7.1. We consider a run consisting of 10000 requests with a warm-up of 1000 requests. Each run is replicated 48 times. Furthermore, we consider a time horizon of 40 time units. Each successful response to a request will gain a reward of $R = 20$ money units while for a failed request $V = 50$ money units has to be paid. All concrete service response-time distributions use the same scaling factors for the low and high response-time means $a^{(i,j)} = 0.85$, $b^{(i,j)} = 2$, and the same variance factor $k^{(i,j)} = 100$ for all $i = 1, \ldots, N$, $j = 1, \ldots, M_i$. Furthermore, we keep the values of $t_{cycle}^{(i,j)}$, $W^{(i,j)}$ and $t_p^{(i,j)}$ fixed for all concrete services. Therefore, we omit index (i,j) for t_{cycle}, W, and t_p in our experiment values. We vary t_{cycle} and W_p in the range $\{10, 15, 20, 50, 100\}$ and vary t_{cycle} in the range $\{200, 500, 1000, 2000, 4000\}$. The parameter effects that we want explore are sample window size, test significance level, test type (e.g. Kolmogorov Smirnov), and probe interval. All experiments has been run until the confidence interval was less than 1%.

Table 7.1: *Concrete service alternatives for task i*

		Parameter		
		c	μ	σ
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	2
	$(\cdot, \mathbf{2})$	2	3	2
	$(\cdot, \mathbf{3})$	5	2.5	2
	$(\cdot, \mathbf{4})$	10	1.25	3

7.6 Results

Using the experimental set-up in Section 7.5.2 we obtained the following performance measures:

- Expected benefit (see Section 7.6.1),
- Number of updates (see Section 7.6.2),
- Number of probes (see Section 7.6.3),
- Average probe cost per request (see Section 7.6.4).

For expected benefit we compare three approaches: (KS) the sliding window approach based on Kolmogorov Smirnov statistic for change detection, (KSS) the smoothing approach with adjusted Kolmorov Smirnov statistic for change detection, and theoretical (TH) from the policy based on the *known in advance* mixture of log-normal distributions. The TH is compared as benchmark where analysis is done on historical data from services and short term changes in response-time distributions are aggregated in to one long term response-time distribution. For all other performance measures we only compare KS and KSS as no updating or probing is involved in the TH approach as the distribution is known/analyzed in advance. However the mixture of log-normal distributions is controlled by a Markov Chain and therefore we expect the KS and KSS approach to perform better.

7.6.1 Expected benefit

Figure 7.4 presents the experimental results on average benefit per request with KS, KSS, and TH as functions of cycle time t_{cycle} (Figures 7.4a, 7.4b), probe time-out t_p (Figure 7.4c), and sliding window sample count W (Figure 7.4d). If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. Note that in Figure 7.4 probe cost is not incorporated and so all probes have cost $c_p^{(i,j)} = 0$, $i = \{1, \dots, N\}$, $j = \{1, \dots, M_i\}$. In Figure 7.4a we observe that (as expected) for short cycle time t_{cycle} KS and KSS perform close to TH. This is because for small t_{cycle} the distribution becomes effectively a mixture of log-normal distributions on which the dynamic program in the TH approach is solved. For very small t_{cycle} the TH approach performs better but this is the case where effectively no change in response-time distributions occurs. Here we are simply comparing learning versus knowing the response-time distributions. For larger t_{cycle} the difference increases as all concrete services are slowly alternating between two distribution states. Figure 7.4c illustrates that a higher probe time-out t_p will decrease expected benefit for KS and KSS. This corresponds to the fact that a higher probe time-out will cause the system to respond slower to changes in less frequently used alternatives which could have become attractive. Also a larger sliding window size W decreases expected benefit for KS and KSS as new response-time observations are smoothed out across more samples.

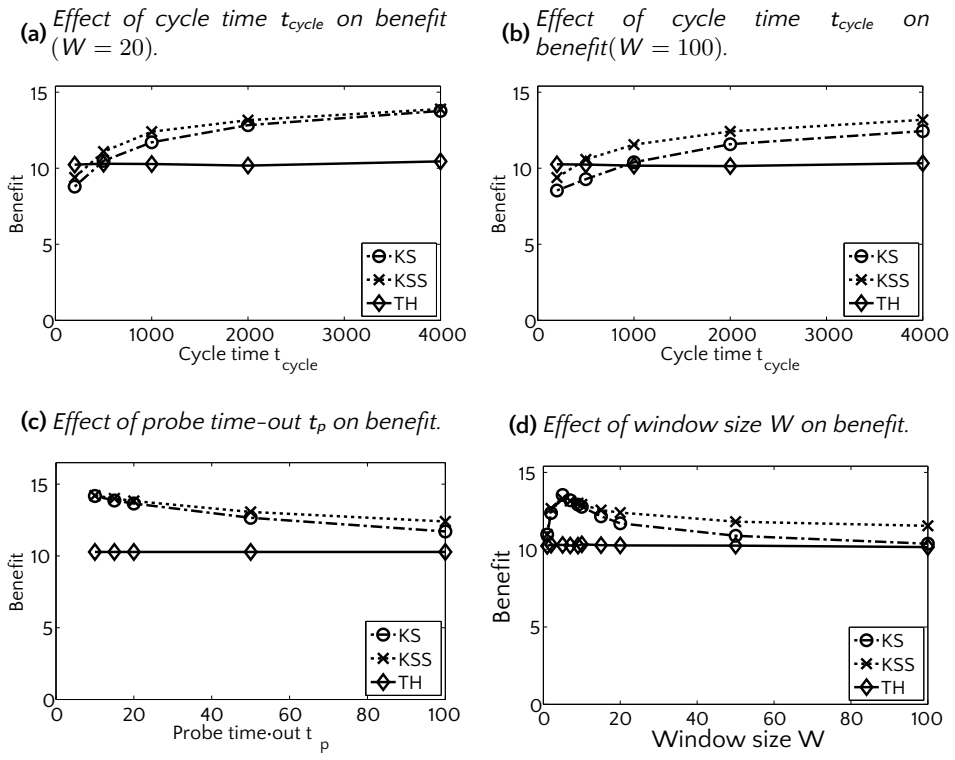


Figure 7.4: *Effect of cycle time, probe time-out and window size on expected benefit per request.*

7.6.2 Number of updates

In the experiments we also recorded the number of lookup table updates. Figure 7.5 contains the required number of lookup-table updates for KS, KSS, and TH as function of cycle time, probe time-out and window size. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. As expected Figure 7.5a illustrates that a larger cycle time corresponds to less updates. This corresponds to the behavior of our experimental set-up that a larger cycle time results in less jumps in the distribution (Markov Chain). Therefore less significant changes should be detected. In Figure 7.5b we observe that the probe time-out has a big impact on the number of lookup table updates. As we update the lookup-table after each probe, the probe time-out should be not too low. In Figure 7.5c we observe quite constant behavior over the window sizes. For large sliding window size the number of updates slightly reduces as the empirical distributions approaches

the actual response-time distributions. However as we observed in Figure 7.4d we obtain a distribution aggregated over a longer time span and are not able to track short term changes in response times.

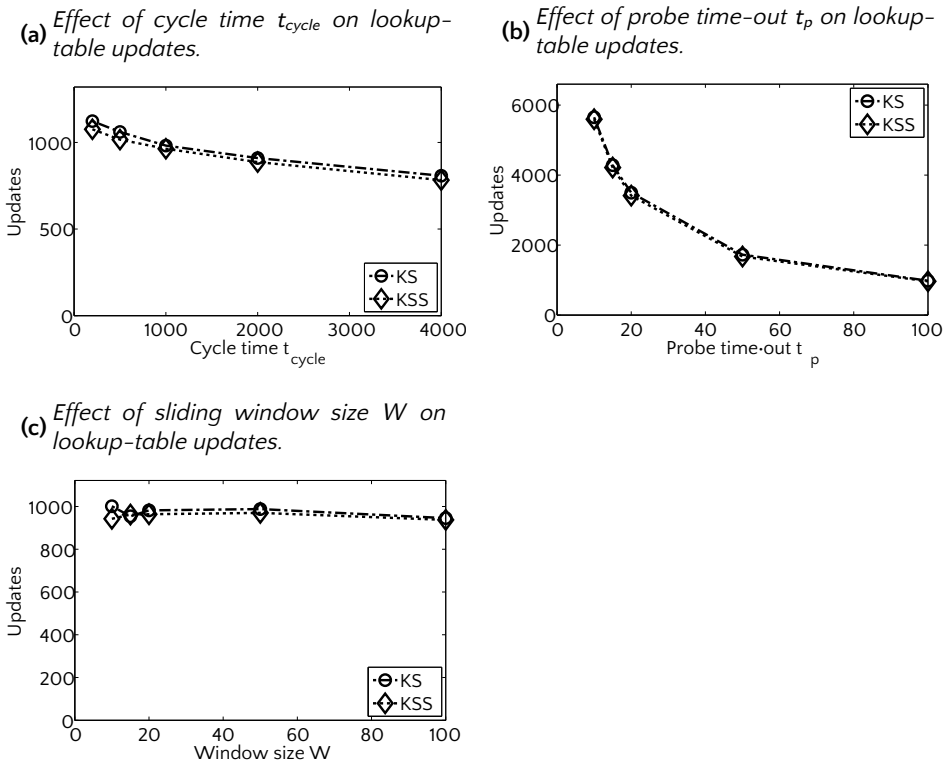


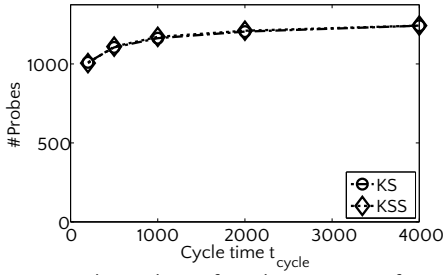
Figure 7.5: *Effect of cycle time, probe time-out and window size on the required number of lookup-table updates.*

7.6.3 Number of probes

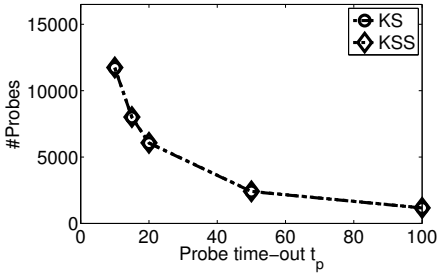
Another interesting measure is the total number of probes that is sent as result of our closed-loop approach. In Figure 7.6 we present the effect of cycle time, probe time-out and window size on the total number of probes sent, for KS, KSS, and TH. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. We observe in Figure 7.6a the interesting behavior that the total number of probes sent increases as the cycle time increases. An explanation for this

behavior is that all concrete service alternative distributions in our experimental set-up are controlled by independent Markov Chains. Therefore, a certain combination of response-time distribution states is observed for a longer time which allows the lookup table to adapt to these specific situations. As a result services that would rarely be invoked are invoked more frequently resulting in less probes (less probe time-outs). As expected we observe in Figure 7.6b that the probe time-out has big effect on the total number of probes sent. This is a result of the probing strategy we proposed in Section 7.3. Figure 7.6c illustrates that the window size has hardly effect on the number of probes sent. A larger window size will make distribution estimates more accurate and will not cause the lookup to change dramatically. So rarely invoked concrete service alternatives are not necessarily visited more frequently as result of a change sliding window size.

(a) *Total number of probes sent as function of cycle time t_{cycle}.*



(b) *Total number of probes sent as function of probe time-out t_p.*



(c) *Total number of probes sent as function of window size W.*

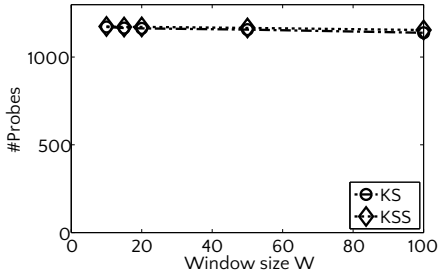


Figure 7.6: *Effect of cycle time, probe time-out and window size on the total number of probes sent.*

7.6.4 Mean probe cost

In Section 7.6.3 we discussed the results on the total number of probes sent for the different algorithms. Alternatively this can be seen as the behavior of probe cost when cost of probing for all concrete service alternatives is equal to 1. In our experiments we also individually tracked the number of probes sent of each concrete service alternative individually. The graphs in Figure 7.7 represent the expected probe cost per request (total cost/#probes) if the probe costs are proportional to the costs of invoking a concrete service alternative: $c_p^{(i,j)} \propto c_p^{(i,j)}$. The graphs represent the case where $c_p^{(i,j)} = c_p^{(i,j)}$, for $i = \{1, \ldots, N\}$, and $j = \{1, \ldots, M_i\}$. For this case we observe similar behavior of the graphs compared to Figure 7.6 in Section 7.6.3. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$.

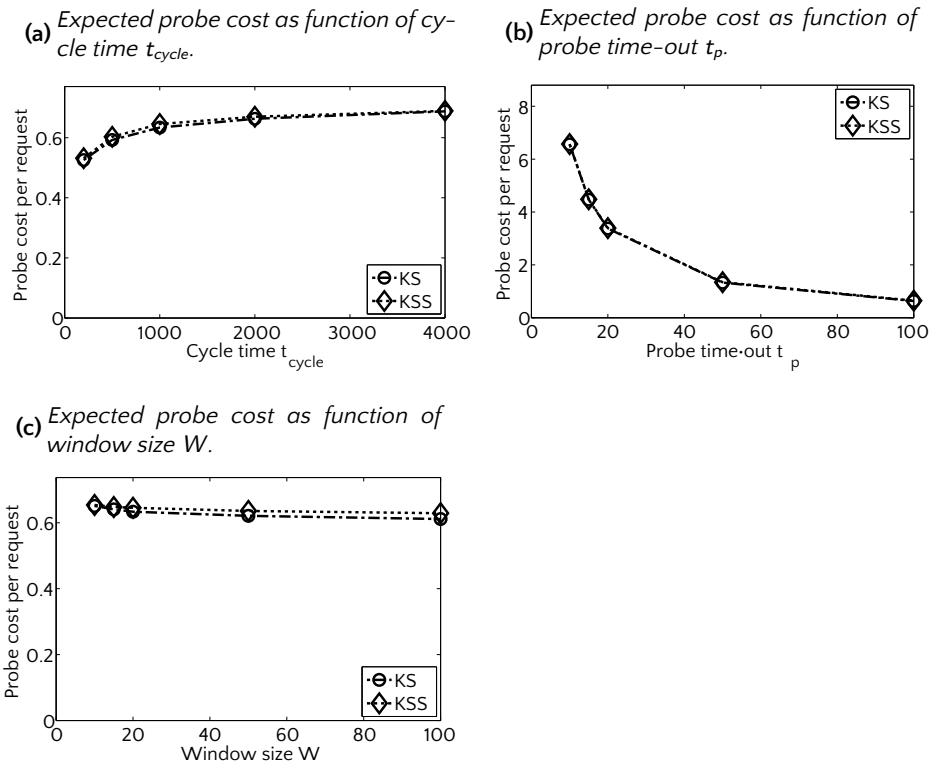


Figure 7.7: *Effect of cycle time, probe time-out and window size on the expected probe cost per request.*

7.7 Discussion

We modeled and implemented a closed-loop approach where dynamic programming is applied on empirical distributions resulting from the actual realized response-time distributions of concrete service providers. Our approach is robust to changes in the sense that it adapts to changes in response-time distributions of concrete service alternatives. To achieve this we use a smoothing approach or sliding window approach on the empirical distribution. The smoothing approach has advantage that there is no overhead in bookkeeping of sliding window samples while past response-time realizations have limited impact in the smoothed empirical distributions. When using our approach there is a trade-off between parameters that we need to optimize. These parameters are the sliding window W or exponential smoothing parameter κ and the change point detection test significance α. The constraints are here determined by computational power and probe cost. Typically we would like to update our lookup table every request and probe frequently. However it takes time to compute a new strategy. Furthermore cost is connected with probes. It would be a waste if a new probe is sent while a previous lookup table computation is still running. We should choose probe time-outs such that we can exploit information about improved service without spending too much cost on probing and using too much computational power. Experimental results indicate that in an environment with changing response-time behavior our closed-loop approach has a significant advantage compared to a static lookup table as our approach has the strong advantage that it learns and exploits response-time behavior on the fly.

Tuning window size W (or corresponding smoothing factor κ) and α creates a second layer of control where these parameters are adapted to optimal values. The update of these parameters is typically on a larger time scale that is not in the scope of our experiments. This is a interesting direction for further research.

Publications of the author

Refereed papers

1. J.W. Bosman and R. Núñez-Queija. A spectral theory approach for extreme value analysis in a tandem of fluid queues. *Queueing Systems*. Accepted for publication in Queueing Systems (subject to minor revision). 2013.

2. J.W. Bosman, G.J. Hoekstra, R.D. van der Mei, and S. Bhulai. A simple index rule for efficient traffic splitting over parallel wireless networks with partial information. *Performance Evaluation*, 70(10):889 - 899. 2013.

3. X. Gao, Y. Lu, M. Sharma, M.S. Squillante, and J.W. Bosman. Stochastic optimal control for a general class of dynamic resource allocation problems. *SIGMETRICS Performance Evaluation Review*, 41(2):3-14. 2013.

4. G.J. Hoekstra, R.D. van der Mei, and J.W. Bosman. Efficient traffic splitting in parallel TCP-based wireless networks: modelling and experimental evaluation. In: *Proceedings of the 25th International Teletraffic Congress, ITC (Shanghai, China, September 2013)*. 2013.

5. S. Bhulai, G.J. Hoekstra, J.W. Bosman, and R.D. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A Bayesian approach. *Performance Evaluation*, 69(1):41-52. 2012.

6. J.W. Bosman, R.D. van der Mei, and R. Núñez-Queija. A fluid model analysis of streaming media in the presence of time-varying bandwidth. In: *Proceedings of the 24th International Teletraffic Congress, ITC (Krakow, Poland, September 2012)*. 2012.

7. M. Živković, J.W. Bosman, J.L. van den Berg, R.D. van der Mei, H.B. Meeuwissen, and R. Núñez-Queija. Run-time revenue maximization for composite Web services with response time commitments. In: *Proceedings of the IEEE 26th International Conference on Advanced Information Networking and Applications conference, AINA (Fukuoka, Japan, March 2012)*, pages 589-596. 2012.

8. M. Živković, J.W. Bosman, J.L. van den Berg, R.D. van der Mei, H.B. Meeuwissen, and R. Núñez-Queija. Dynamic profit optimization of composite Web services with SLAs. In: *Proceedings of the IEEE Global Telecommunications conference, GlobeCom (Houston, TX, December 2011)*, pages 1-6. 2011.

9. G.J. Hoekstra, R.D. van der Mei, and J.W. Bosman. On comparing the performance of dynamic multi-network optimizations. In: *Proceedings of the IEEE Global Telecommunications Conference, GlobeCom (Miami, FL, December 2010)*, pages 1–5. 2010.

Submitted papers

10. J.W. Bosman, J.L. van den Berg, and R.D. van der Mei. Autonomous runtime QoS control for composite services in SOA.

11. M. Živković, J.W. Bosman, J.L. van den Berg, and R.D. van der Mei. Profit maximization with dynamic service selection in SOA.

12. L. Duijvestijn, J.W. Bosman, R.D. van der Mei, H.B. Meeuwissen, and M. Živković. A QoS control framework for real-time orchestration of composite services.

Summary

Optimal QoS control in Communication Systems

In current practice, quality of composite services is usually controlled on an ad-hoc basis, while the consequences of failures in service chains are often not well understood. A main concern is that, although such an approach might work for small chains, it will become unfeasible for future complex global-scale service chains. This raises the need for mechanisms that enable efficient usage of available shared resources while preserving the desired Quality of Service (QoS) as perceived by the end user. There are many optimization mechanisms available that could accomplish this. The problem is that in general these mechanisms are not suitably tailored for the current and evolving information and communication systems. The controls and thresholds are often based on simple improvised rules. As a consequence, the enormous potential of QoS mechanisms to enhance service quality remains largely unexploited.

The main challenge that is faced in this dissertation is: *how to effectively use QoS mechanisms for large-scale complex ICT systems with shared resources.*

To this end, we develop, analyze, optimize and evaluate quantitative models that capture the dynamics of QoS-control mechanisms and their implications on the user-perceived QoS. The development of efficient QoS mechanisms is complicated by the omnipresence of the phenomenon of uncertainty. Stochastic models are instrumental to capture such uncertainties and provide a basis for educated control of systems with uncertainty. One may distinguish the following three types of uncertainty.

Uncertainty about demand for resources. An inportant deal of demand for resources is driven by predictable user behavior. However, there are also many factors that are inherently unpredictable but may have a huge impact on resource availability (cyber attacks, flash crowds). For this purpose, mechanisms are required that can respond to this unpredictable behavior and provide robustness to threats and undesired behavior.

Variability in resource availability (shared resources). Various factors contribute to variability in resource availability such as resource sharing, network or system failure, chaotic behavior, and temporary overload. For a majority of Internet resources, capacity is shared among the different users. As a result, in the perspective of the users, the availability of resource capacity varies. Another contributing factor to variablity that may need explanation here is chaotic behavior. Chaotic behavior may for

example be caused by unexpected interactions between systems, often due to misconfiguration. In worst cases misconfiguration causes network or system failures. This is especially the case for (global) systems where demand volumes are so high that individual systems cannot handle all demand.

Limited information. Many existing models assume that the stochastic behavior of demand and resources is known. In practice, however this is rarely the case. Typically external parties at best have limited information about the internal behavior of a system. Also external factors impact the challenge of limited information from system behavior. Systems possibly operate in changing environments driven by uncertain, unpredictable factors. To respond in a fashionable way, mechanisms are required that can adapt to these changes.

Over the past few years, the tremendous popularity of smart mobile end devices and services (like YouTube) has boosted the demand for streaming media applications offered via the Internet. As the Internet provides no more than best-effort service quality, packet streams generated by streaming media applications are distorted by fluctuations in the available bandwidth, which may be significant over the duration of a typical streaming application. To cope with these distortions, play-out buffers temporarily store packets so as to reproduce the signal with a fixed delay offset. In Chapters 2 and 3 we study a video stream model where the network is modeled as a Markov Modulated fluid queue. In this model a Contineous Time Markov Chain represents the actual transmission rate through the network. Chapter 2 considers a two-state transmission rate model while Chapter 3 considers a more general transmission rate model. For the play-out buffer an initial buffer level b_{init} is determined such that the probability that the video will stall during play-out will not exceed an agreed service level probability p_{empty}. We show that the probability of this event corresponds to the probability of the event where the maximum congestion level $M(t)$ exceeds the initial buffer level b_{init}. From this insight we derive an expression that maps p_{empty}, T_{play} and the network and video parameters to a minimal buffer level b_{init}. Simulation results indicate that the buffer level that is obtained from our analysis is a conservative estimate, i.e., it overestimates the true minimal required buffer level.

In Chapter 4 we consider the transmission of file flows across multiple parallel wireless networks. Each wireless network is modeled as a processor sharing node. In this setting background flows are generated by clients with only one available network connection while foreground flows are generated by clients with multiple network connections. The goal is to minimize the expected transfer time of elastic data traffic by smartly dispatching the jobs of foreground flows to the networks. However only partial information is available in the sense that only the sum of the numbers of foreground and background flows can be observed. To this end, we propose a simple index rule called the convex combination (CC) rule. Extensive simulations with real networks show that this method performs extremely well under practical cir-

cumstances for a wide range of realistic parameter settings. The method presented in this chapter is a simple index rule that is essentially a convex combination of techniques that are found to work well extreme cases. To assess the effectiveness of the CC method, we have performed extensive simulation experiments in a real network simulator that implements the full wireless protocols stack. The results show that the CC method leads to close-to-optimal performance for a wide range of realistic parameter settings.

In Chapter 5 we investigate a general class of dynamic resource allocation problems that involve different types of resources and uncertain/variable demand. Aiming to maximize the expected net-benefit based on rewards and costs from the different resources, an optimal dynamic control policy has been derived within a singular stochastic optimal control setting. The mathematical analysis includes obtaining simple expressions that govern the dynamic adjustments to resource allocation capacities over time under the optimal control policy. Based on this analysis, a wide variety of extensive numerical experiments have been constructed. The results demonstrate and quantify significant benefits of the optimal dynamic control policy over recently proposed alternative optimization approaches in addressing a general class of resource allocation problems across a diverse range of application domains. Moreover, our results strongly suggest that the approach taken in this chapter can provide an effective means to develop easily-implementable online algorithms for solving stochastic optimization problems.

In Chapter 6 we address dynamic decision mechanisms for composite web services. We represent the composite web-service as a (sequential) workflow of tasks. For each task within this workflow, a number of third-party service alternatives may be available, offering the same functionality at different price-quality levels. Before a task in the workflow can be executed, a service alternative must be selected that implements the task functionality. We have developed a model to maximize benefit for composite services by on-the-fly dynamic service selection. The selection decisions are based on observed response times, the response-time characteristics of the alternative, the end-to-end response-time objectives, and the reward and penalty parameters. The results not only indicate *that* there is an enormous potential gain compared to other, non-dynamic approaches, but also show *how* one can realize such gains. We believe that this work is a significant step in realizing cost-efficient provisioning of complex composite services.

In Chapter 7 we propose a runtime closed-loop control mechanism that dynamically optimizes service composition in real time by learning and adapting to changes in third party service response time behaviors. We extend the dynamic programming approach of Chapter 6 to a closed-loop approach where dynamic programming is applied on empirical distributions resulting from the actual realized response-times of third party service providers. Our approach is robust to changes in the sense that it adapts to changes in response-time distributions of concrete service alternatives.

To achieve this we use a smoothing approach or a sliding window approach on the empirical distribution. The smoothing approach has the advantage that there is no overhead in bookkeeping of sliding window samples. When using our approach must strike a balance between parameters that we use in the optimization such as the sliding window W or exponential smoothing parameter κ and the change point detection test significance α. These parameter values are constrained by computational power and probe cost. Experimental results indicate that in an environment with changing response-time behavior our closed-loop approach has a significant advantage as it learns and exploits response-time behavior on the fly compared to a static lookup table that does not account for environment changes.

Samenvatting (Dutch Summary)

Optimale besturing van serviceniveaus in ICT-systemen

De hedendaagse, complexe, samengestelde ICT-systemen worden vaak op een ongecentraliseerde wijze bestuurd zonder dat er een goed inzicht is in de gevolgen van storingen in *ketens* van ITC-diensten. Hoewel deze aanpak goed kan werken voor kleine ketens, wordt dit onhaalbaar voor complexe wereldwijde dienstenketens. Daarom zijn er mechanismen nodig die op een efficiënte wijze beschikbare (netwerk)systeemcapaciteit kunnen benutten zonder aan het gewenste serviceniveau voor de eindgebruikers in te boeten.

Het belangrijkste vraagstuk dat in deze dissertatie wordt behandeld is: *hoe kunnen serviceniveaumechanismen effectief worden toegepast op complexe grootschalige ICT-systemen met gedeelde systeemcapaciteit?*

Om deze vraag te beantwoorden worden er in deze dissertatie kwantitatieve modellen ontwikkeld, geanalyseerd, geoptimaliseerd en geëvalueerd die de essentiële dynamiek beschrijven van op service gerichte besturingsmechanismen en wordt het gevolg bestudeerd van het gebruik van deze modellen op het (door de gebruikers ervaren) serviceniveau. Het achterliggende doel van deze aanpak is om schaalbare, robuuste algoritmen, beslistabellen en vuistregels te ontwikkelen die het mogelijk maken om service gerichte besturingsmechanismen optimaal toe te passen. Bij de toepassing hiervan worden drie complicerende factoren onderscheiden:

Veranderlijkheid van de vraag naar systeemcapaciteit. Een groot deel van de vraag naar systeemcapaciteit wordt bepaald door voorspelbaar gedrag van gebruikers. Naast voorspelbaar gedrag zijn er ook moeilijk te voorspellen fenomenen die een grote invloed hebben op de beschikbaarheid van systeembronnen, zoals aanvallen via het internet of onverwachte drukte door bijvoorbeeld het bekend worden van een grote gebeurtenis in de media.

Onzekerheid over de beschikbaarheid van systemen. Verschillende zaken dragen bij aan de variatie in beschikbaarheid van systemen. Voorbeelden hiervan zijn het delen van systeemcapaciteit, uitval van netwerken en systemen, chaotisch gedrag van systemen en tijdelijke overbelasting. In de meeste ICT-systemen is het delen van systeemcapaciteit de belangrijkste bron van variabiliteit in de beschikbaarheid. Een andere belangrijke factor is chaotisch gedrag van systemen als gevolg van een onverwachte wisselwerking tussen verschillende systemen. Vaak wordt dergelijk gedrag veroorzaakt door configuratiefouten. In het uiterste geval kunnen netwerken en systemen vastlopen. Dit is in het bijzonder het geval voor globale systemen waar

de vraagvolumes dusdanig groot zijn dat losse systemen niet in staat zijn om alle vraag individueel af te handelen.

Beperkte informatie over wat zich afspeelt in externe systemen. Veel gebruikte modellen veronderstellen dat het stochastische gedrag van vraag en systeemcomponenten of capactiteit bekend is. In de praktijk is dit zelden het geval. Meestal hebben gebruikers slechts beperkt zicht op wat er zich afspeelt in de systemen van externe partijen die zij gebruiken. Bovendien is het mogelijk dat de systemen draaien in een omgeving die onderhevig is aan onzekere en onvoorspelbare factoren. Om met die beperkte informatie om te kunnen gaan, zijn mechanismen nodig die zich kunnen aanpassen aan deze onzekere en veranderende factoren.

Hoofdstukken 2 en 3 beschouwen een vloeistofmodel dat het gedrag van video over het internet, bijvoorbeeld YouTube, beschrijft. Een storende factor in video's over het internet is dat deze kunnen gaan haperen tijdens het afspelen. Uit het vloeistofmodel volgt een aanpak waarmee de laadtijd en andere parameters zoals videokwaliteit en bandbreedte zo kunnen worden gekozen dat een video met een grote waarschijnlijkheid onafgebroken afspeelt.

In hoofdstuk 4 wordt de situatie beschouwd waarin bestanden kunnen worden verstuurd over meerdere draadloze netwerken. Voor deze situatie wordt een eenvoudig toepasbare beslisregel geformuleerd, genaamd convexe combinatie (CC), die bepaalt over welk netwerk een bestand moet worden verstuurd, gebruikmakend van de geobserveerde drukte in de netwerken. De beslisregel is gebaseerd op een combinatie van twee regels die goed werken in verschillende situaties. Om de effectiviteit van onze beslisregel te evalueren is de CC regel geïmplementeerd in een simulatieomgeving die het gedrag van netwerken realistisch nabootst. Uit de resultaten blijkt dat de CC regel goed presteert bij een breed scala aan belastings- en capaciteitsparameters van draadloze netwerken.

In hoofdstuk 5 wordt een toewijzingsprobleem behandeld. Er zijn twee diensten aanwezig: een interne (goedkopere) dienst en een (duurdere) dienst van een externe partij. Verder is er een variërend vraagproces dat zowel lange termijn patronen vertoond als korte termijn schommelingen. De uitdaging is om de interne capaciteit goed te kiezen, zodat de schommelingen kunnen worden opgevangen. Indien er meer vraag is dan toegewezen interne capaciteit gaat er vraag verloren. In het geval dat teveel capaciteit is toegewezen, wordt er betaald voor ongebruikte capaciteit. Echter, aan het aanpassen van de interne capaciteit zijn ook kosten verbonden. Het is van belang om een goede afweging te maken tussen aanpassingskosten van de interne capaciteit en de door schommelingen in het vraagproces veroorzaakte onder- of overcapaciteit. In dit hoofdstuk wordt een eenvoudig te implementeren mechanisme geformuleerd dat goed met deze schommelingen om kan gaan. Om dit mechanisme te demonstreren is er een simulatieomgeving opgezet. Uit de experimenten blijkt dat het beschreven besturingsmechanisme zeer goed functioneert.

Hoofdstuk 6 beschouwt dynamische compositie van samengestelde webdiensten. Daarbij wordt de samengestelde webdienst als een keten van taken gerepresenteerd die sequentiëel moeten worden uitgevoerd. Voor elke taak in de keten zijn implementaties beschikbaar van externe partijen met elk hun eigen prijs-kwaliteitsverhouding. De samengestelde webdienst is onderdeel van een service-overeenkomst waarin staat dat de respons op elke vraag binnen een vastgestelde termijn plaats moet vinden. Met deze overeenkomst voor ogen is in dit hoofdstuk een algoritme ontwikkeld dat een dynamische beslisstrategie berekent voor de gegeven serviceovereenkomst en prijs-kwaliteitsverhouding van de gebruikte diensten van externe partijen. Het algoritme neemt beslissingen op basis van de resterende responstijd voordat het in de serviceovereenkomst gestelde tijdsdoel wordt overschreden. Uit experimenten blijkt dat er enorme winst valt te behalen door de compositie dynamisch te laten aanpassen aan de resterende responstijd.

In hoofdstuk 7 wordt uitgegaan van de dynamische beslisstructuur van hoofdstuk 6. Echter, dit maal wordt er verondersteld dat het gedrag in termen van responstijd van derde partijen niet bekend is en geleerd moet worden uit geobserveerde responstijden. Dit hoofdstuk een ontwikkelt aanpak waarbij een dynamische programmeertechniek wordt toegepast die is gebaseerd op de *empirische responstijdverdelingen*. De empirische responstijdverdelingen worden actueel gehouden door middel van twee mogelijke principes het vensterprincipe en het uitdoofprincipe. Op de empirische verdelingen worden statistische toetsen toegepast om te kijken of er significante veranderingen zijn geweest in de responstijdverdelingen. Op deze manier hoeft het dynamisch programmeeralgortime niet voor elke waarneming een nieuwe beslistabel te berekenen. Om te voorkomen dat bepaalde diensten nooit bezocht worden, omdat deze diensten niet in de dynamische beslistabel zitten. Mogelijk vormen deze onbezochte diensten toch een aantrekkelijk alternatief, omdat ze beter zijn gaan presteren. Daarom worden er testaanvragen verstuurd. Dit zijn aanvragen die informatie inwinnen over de onbezochte diensten. Om de beschreven aanpak goed te laten werken moeten verschillende parameters worden afgewogen. Uit de simulatie-experimenten blijkt dat de beschreven aanpak in veranderende omgevingen veel winst kan opleveren ten opzichte van statische aanpakken die worden berekend over een langere termijn.

Bibliography

[1] M. Abundo, V. Cardellini, and F. Lo Presti. An MDP-based admission control for Service-Oriented Systems. *DISP, Univ. of Roma" Tor Vergata", Tech. Rep. RR-11.86*. 2011.

[2] S.C. Albright. Structural results for partially observable Markov Decision Processes. *Operations Research*, 27:1041-1053. 1979.

[3] H. Amur, J. Cipar, V. Gupta, Gregory R. Ganger, M.A. Kozuch, and K. Schwan. Robust and flexible power-proportional storage. In: *Proceedings of the 1st ACM symposium on Cloud computing, SoCC (Indianapolis, IN, June 2010)*, SoCC '10, pages 217-228. ACM, New York, NY, USA. 2010.

[4] S. Asmussen. Busy period analysis, rare events and transient behavior in fluid flow models. *Journal of Applied Mathematics and Stochastic Analysis*, 7(3):269-299. 1994.

[5] S. Asmussen. Extreme value theory for queues via cycle maxima. *Extremes*, 1(2):137-168. 1998.

[6] S. Asmussen and M. Bladt. A sample path approach to mean busy periods for Markov-modulated queues and fluids. *Advances in applied probability*, pages 1117-1121. 1994.

[7] H. Bannazadeh and A. Leon-Garcia. Online optimization in application admission control for service oriented systems. In: *Proceedings of the IEEE Asia-Pacific Services Computing Conference, APSCC '08 (Yilan, Taiwan, December 2008)*, pages 482-487. 2008.

[8] A.G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341-379. 2003.

[9] J.V.L. Beckers, I. Hendrawan, R.E. Kooij, and R.D. van der Mei. Generalized processor sharing models for Internet access lines. In: *Proceedings of the IFIP Conference on Performance Modelling and Evaluation of ATM and IP networks (Budapest, Hungary, June 2001)*, pages 101-112. Budapest. 2001.

[10] R.E. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press. 1961.

[11] R.E. Bellman. *Dynamic Programming*. Dover Books on Mathematics. Dover. 2003.

[12] V.E. Beneš, L.A. Shepp, and H.S. Witsenhausen. Some solvable stochastic control problems. *Stochastics*, 4(1):39–83. 1980.

[13] S.M. Berman. Limiting distribution of the maximum term in sequences of dependent random variables. *The Annals of mathematical statistics*, 33(3):894–908. 1962.

[14] S. Bhulai, G.J. Hoekstra, J.W. Bosman, and R.D. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A Bayesian approach. *Performance Evaluation*, 69(1):41–52. 2012.

[15] N. Bléfari-Melazzi, V. Eramo, and M. Listanti. Dimensioning of play-out buffers for real-time services in a B-ISDN. *Computer Communications*, 21(11):980 – 995. 1998.

[16] K. Boloor, R. Chirkova, T. Salo, and Y. Viniotis. Analysis of response time percentile service level agreements in SOA-based applications. In: *Proceedings of the IEEE Global Telecommunications Conference, GlobeCom (Houston, TX, December 2011)*, pages 1–6. 2011.

[17] V.A. Bolotin, Y. Levy, and D. Liu. Characterizing data connection and messages by mixtures of distributions on logarithmic scale. In: *Proceedings of the 16th International Teletraffic Congress, ITC (Edinburgh, UK, June 1999)*, pages 887–894. 1999.

[18] S.C. Borst, O.J. Boxma, and N. Hegde. Sojourn times in finite-capacity Processor-Sharing queues. In: *Proceedings of the 1st Conference on Next Generation Internet Networks Traffic Engineering, NGI (Rome, Italy, April 2005)*. 2005.

[19] J.W. Bosman, G.J. Hoekstra, R.D. van der Mei, and S. Bhulai. A simple index rule for efficient traffic splitting over parallel wireless networks with partial information. *Performance Evaluation*, 70(10):889 – 899. 2013.

[20] J.W. Bosman and R. Núñez-Queija. A spectral theory approach for extreme value analysis in a tandem of fluid queues. *Queueing Systems*. Accepted for publication in Queueing Systems (subject to minor revision). 2013.

[21] J.W. Bosman, J.L. van den Berg, and R.D. van der Mei. Autonomous runtime QoS control for composite services in SOA. Submitted for publication.

[22] J.W. Bosman, R.D. van der Mei, and R. Núñez-Queija. A fluid model analysis of streaming media in the presence of time-varying bandwidth. In: *Proceedings of the 24th International Teletraffic Congress, ITC, (Krakow, Poland, September 2012)*. Krakow, Poland. September 2012.

[23] O.J. Boxma and V. Dumas. The busy period in the fluid queue. 26(1). 1998.

[24] R.I. Brafman. A heuristic variable grid solution method for POMDPs. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence (Providence, RI, July 1997)*, pages 727–733. 1997.

[25] A.N. Burnetas and M.N. Katehakis. Optimal adaptive policies for Markov Decision Processes. *Mathematics of Operations Research*, 22:222–255. 1997.

[26] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In: *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1069–1075. ACM. 2005.

[27] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani. A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software*, 81(10):1754–1769. 2008.

[28] V. Cardellini, E. Casalicchio, V. Grassi, and F. Lo Presti. Adaptive management of composite services under percentile-based Service Level Agreements. In: *Proceedings of the 8th International Conference on Service-Oriented Computing, ICSOC (San Francisco, CA, December, 2010)*, volume 6470, page 381. Springer-Verlag New York Inc. 2010.

[29] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281 - 308. 2004.

[30] A.R. Cassandra. *Exact and approximate algorithms for Partially Observable Markov Decision Processes*. Ph.D. thesis, Brown University. 1998.

[31] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In: *Proceedings of the The 23rd Conference of the IEEE Communications Society, INFOCOM (Hong Kong, China, March 2004)*. 2004.

[32] F. Chen, D. Lambert, and J.C. Pinheiro. Incremental quantile estimation for massive tracking. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD (Boston, MA, August 2000)*, pages 516–522. 2000.

[33] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive Internet services. In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI (San Fransisco, CA, April 2008)*, volume 8, pages 337–350. 2008.

[34] G.L. Choudhury and D.J. Houck. Combined queuing and activity network based modeling of sojourn time distributions in distributed telecommunication systems. *The Fundamental Role of Teletraffic in the Evolution of Telecom-*

munications Networks (Eds. J. Labetoulle and JW Roberts), Proceedings of ITC 14, 14:525–534. 1994.

[35] D.F. Ciocan and V. Farias. Model predictive control for dynamic resource allocation. *Mathematics of Operations Research*, 37(3):501–525. 2012.

[36] Cisco. Visual Networking Index: Forecast and Methodology, 2012–2017. Cisco white paper, Cisco. 05 2013.

[37] Cisco. Visual Networking Index: Global Mobile Data Traffic Forecast Update 2012–2017. Cisco white paper, Cisco. 02 2013.

[38] J.F. Claerbout. *Fundamentals of geophysical data processing*. Pennwell Books, Tulsa, OK. 1985.

[39] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, NSDI (Boston, MA, May 2005)*, volume 2 of *NSDI'05*, pages 273–286. USENIX Association, Berkeley, CA, USA. 2005.

[40] D. Cox. Fundamental limitations on the data rate in wireless systems. *IEEE Communications Magazine*, 46(12):16–17. 2008.

[41] A. Dua and N. Bambos. Buffer Management for Wireless Media Streaming. In: *Proceedings of the IEEE Global Telecommunications Conference, GlobeCom (Washington, DC, November 2007)*, pages 5226–5230. 2007.

[42] L. Duijvestijn, J.W. Bosman, R.D. van der Mei, H.B. Meeuwissen, and M. Živković. A QoS control framework for real-time orchestration of composite services. Submitted for publication.

[43] J. Duncanson. Inverse multiplexing. *IEEE Communications Magazine*, 32(4):34–41. 1994.

[44] R. El-Yaniv, R. Kaniel, and N. Linial. Competitive optimal on-line leasing. *Algorithmica*, 25(1):116–140. 1999.

[45] E.O. Elliott. Estimates of error rates for codes on burst-noise channels. *Bell System Technical Journal*, 42:1977–1997. September 1963.

[46] FCC. Report of the spectrum efficiency working group. Technical report, Federal Communications Commission Spectrum Policy Task Force. November 2002.

[47] F.R. Gantmacher. *Matrix Theory vol. 1*. AMS Chelsea Publishing. 2000.

[48] X. Gao, Y. Lu, M. Sharma, M.S. Squillante, and J.W. Bosman. Stochastic optimal control for a class of dynamic resource allocation problems. Technical report, IBM Research Div. 2012.

[49] X. Gao, Y. Lu, M. Sharma, M.S. Squillante, and J.W. Bosman. Stochastic optimal control for a general class of dynamic resource allocation problems. *SIGMETRICS Performance Evaluation Review*, 41(2):3–14. 2013.

[50] S. Ghosh, J. Kalagnanam, D. Katz, M. Squillante, and Xiaoxuan Zhang. Integration of demand response and renewable resources for power generation management. In: *Proceedings of the IEEE PES conference on Innovative Smart Grid Technologies, ISGT (Berlin, Germany, October 2012)*, pages –. 2011.

[51] E.N. Gilbert et al. Capacity of a burst-noise channel. *Bell Syst. Tech. J*, 39(9):1253–1265. 1960.

[52] C. Gkantsidis, M. Ammar, and E. Zegura. On the effect of large-scale deployment of parallel downloading. In: *Proceedings of the Third IEEE Workshop on Internet Applications, WIAPP (San Jose, CA, June 2003)*, page 79. IEEE Computer Society, Washington, DC, U.S.A. 2003.

[53] A. Gosavi. Reinforcement learning: a tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192. 2009.

[54] The Multipath TCP (MPTCP) working group. Multipath TCP (mptcp) charter. `http://datatracker.ietf.org/wg/mptcp/charter/`. April 2011.

[55] B. Guenter, N. Jain, and C. Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In: *Proceedings of the 30th IEEE International Conference on Computer Communications, INFOCOM (Shanghai, China, April 2011)*, pages 1332–1340. 2011.

[56] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Deployable multipath communication scheme with sufficient performance data distribution method. *Computer Communications*, 30(17):3285–3292. 2007.

[57] M. Hauskrecht. *Planning and Control in Stochastic Domains with Imperfect Information*. Ph.D. thesis, Massachusetts Institute of Technology. 1997.

[58] G.J. Hoekstra and F.J.M. Panken. Increasing throughput of data applications on heterogeneous wireless access networks. In: *Proceedings of the 12th IEEE Symposium on Communication and Vehicular Technology in the Benelux, SCVT (Twente, The Netherlands, 2005)*. 2005.

[59] G.J. Hoekstra and R.D. van der Mei. Effective load for flow-level performance modelling of file transfers in wireless LANs. *Computer Communications*, 33(16):1972–1981. 2010.

[60] G.J. Hoekstra, R.D. van der Mei, and J.W. Bosman. On comparing the performance of dynamic multi-network optimizations. In: *Proceedings of the IEEE Global Telecommunications Conference, GlobeCom (Miami, FL, December 2010)*, pages 1–5. 2010.

[61] G.J. Hoekstra, R.D. van der Mei, and J.W. Bosman. Efficient traffic splitting in parallel TCP-based wireless networks: modelling and experimental evaluation. In: *Proceedings of the 25th International Teletraffic Congress, ITC (Shanghai, China, September 2013)*. 2013.

[62] H.Y. Hsieh and R. Sivakumar. A Transport Layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1):99-114. 2005.

[63] S. Hwang, H. Wang, J. Tang, and J. Srivastava. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Information Sciences*, 177(23):5484 - 5503. A selection of the very best extended papers of the IMS-2004 held at Sarkaya University in Turkey. 2007.

[64] IEEE Standard 802.11n. Part 11: Wireless LAN Medium Access Control (MAC) and physical layer specifications enhancements for higher throughput. October 2009.

[65] D.L. Iglehart. Extreme values in the GI/G/1 queue. *The Annals of Mathematical Statistics*, 43(2):627-635. 1972.

[66] M.C. Jaeger, G. Rojec-Goldmann, and G. Muhl. QoS aggregation for Web service composition using workflow patterns. In: *Proceedings of the 14th IEEE International Enterprise Distributed Object Computing Conference, EDOC (Vitória, Brazil, October 2010)*, pages 149-159. 2004.

[67] K. Jagannathan, I. Menache, E. Modiano, and G. Zussman. Non-cooperative spectrum access - The dedicated vs. free spectrum choice. *IEEE Journal on Selected Areas in Communications*, 30(11):2251-2261. 2012.

[68] I. Karatzas and S. E. Shreve. *Methods of mathematical finance*, volume 39. Springer-Verlag. 1998.

[69] I. Karatzas and S.E. Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer-Verlag, Second edition edition. 1991.

[70] Taehyun Kim, N. Avadhanam, and S. Subramanian. Dimensioning Receiver Buffer Requirement for Unidirectional VBR Video Streaming over TCP. In: *Proceedings of the International Conference on Image Processing, ICIP (Atlanta, GA, October 2006)*, pages 3061-3064. 2006.

[71] K.P. Kontovassilis, J.T. Tsiligaridis, and G.I. Stassinopoulos. Buffer dimensioning for delay- and loss-sensitive traffic. *Computer Communications*, 18(5):315 - 328. 1995.

[72] G.P. Koudouris, R. Agüero, E. Alexandri, J. Choque, K. Dimou, H.R. Karimi, H. Lederer, J. Sachs, and R. Sigle. Generic link layer functionality for multi-

radio access networks. In: *Proceedings of the 14th IST Mobile and Wireless Communications Summit (Dresden, Germany, June 2005)*. 2005.

[73] N.V. Krylov. *Controlled diffusion processes*, volume 14. Springer-Verlag. 1980.

[74] V.G. Kulkarni. Fluid models for single buffer systems. *Frontiers in queueing: Models and applications in science and engineering*, pages 321–338. 1997.

[75] V.G. Kulkarni and E. Tzenova. Mean first passage times in fluid queues. *Operations Research Letters*, 30(5):308–318. 2002.

[76] P.R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23:329–380. 1985.

[77] P. Leitner. Ensuring cost-optimal SLA conformance for composite service providers. In: *ICSOC/ServiceWave 2009 PhD Symposium*, page 43. 2009.

[78] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Schaling up. In: *Proceedings of the twelfth International Conference on Machine Learning, ICML (Tahoe City, CA, July, 1995)*, pages 362–370. 1995.

[79] J.A. Loeve. *Markov Decision Chains with Partial Information*. Ph.D. thesis, Leiden University. 1995.

[80] W.S. Lovejoy. A survey of algorithmic methods for Partially Observed Markov Decision Processes. *Annals of Operations Research*, 28:47–66. 1991.

[81] A. Mahajan and D. Teneketzis. Multi-armed bandit problems. In: *Foundations and Applications of Sensor Management*, pages 121–151. Springer-Verlag. 2007.

[82] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, Internet Engineering Task Force. 1996.

[83] L. Minghong, A. Wierman, L.L.H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In: *Proceedings of the 30th IEEE International Conference on Computer Communications, INFOCOM (Shanghai, China, April 2011)*, pages 1098–1106. 2011.

[84] Farid Molazem Tabrizi, Joseph Peters, and Mohamed Hefeeda. Dynamic Control of Receiver Buffers in Mobile Video Streaming Systems. *Mobile Computing, IEEE Transactions on*, 12(5):995–1008. 2013.

[85] G.E. Monahan. A survey of Partially Observable Markov Decision Processes: theory, models, and algorithms. *Management Science*, 28:1–16. 1982.

[86] J.R. Norris. *Markov chains*. 2008. Cambridge university press. 1998.

[87] OPNET Technologies Inc. OPNET Modeler. `http://www.opnet.com/solutions/network_rd/modeler.html`. November 2011.

[88] G. Pacifici, M. Spreitzer, A.N. Tantawi, and A. Youssef. Performance management for cluster-based web services. *IEEE Journal on Selected Areas in Communications*, 23(12):2333–2343. 2005.

[89] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450. 1987.

[90] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1088–1094. 1995.

[91] H. Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer. 2009.

[92] S.S. Pillai and N.C. Narendra. Optimal replacement policy of services based on Markov Decision Process. In: *Proceedings of the IEEE International Conference on Services Computing, SCC (Bangalore, India, September 2009)*, pages 176–183. 2009.

[93] C. Preist. A conceptual architecture for semantic web services. *The Semantic Web–ISWC 2004*, pages 395–409. 2004.

[94] M.L. Puterman. *Markov Decision Processes: discrete stochastic dynamic programming*. John Wiley & Sons. 1994.

[95] P. Rodriguez, A. Kirpal, and E. Biersack. Parallel-access for mirror sites in the Internet. In: *Proceedings of the IEEE Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, INFOCOM (Tel Aviv, Israel, March 2000)*, pages 864–873. 2000.

[96] S. Rosario, A. Benveniste, S. Haar, and C. Jard. Probabilistic QoS and soft contracts for transaction-based Web services orchestrations. *IEEE Transactions on Services Computing*, 1(4):187–200. 2008.

[97] D. Sarkar, P.D. Amer, and R. Stewart. Guest Editorial: Concurrent multipath transport. *Computer Communications*, 30(17):3215–3217. 2007.

[98] W.R.W. Scheinhardt. *Markov-modulated and feedback fluid queues*. Ph.D. thesis, Faculty of Mathematical Sciences, University of Twente, Enschede, The Netherlands, 1998, http://www. ub. utwente. nl/webdocs/tw/1/t0000008. pdf. 1998.

[99] W.R.W. Scheinhardt and A.P. Zwart. A tandem fluid queue with gradual input. *Probability in the Engineering and Informational Sciences*, 16(1):29–45. 2002.

[100] B. Sengupta and D.L. Jagerman. A conditional response time of the M/M/1 processor-sharing queue. *AT&T Technical Journal*, 64(2):409–421. 1985.

[101] B. Sericola and M.A. Remiche. Maximum level and hitting probabilities in stochastic fluid flows using matrix differential riccati equations. *Methodology and Computing in Applied Probability*, 13(2):307–328. 2011.

[102] A.W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press. 1998.

[103] K.M. van Hee. *Bayesian control of Markov Chains*. Ph.D. thesis, Technical University of Eindhoven. 1978.

[104] P.P. Varaiya, F.F. Wu, and J.W. Bialek. Smart operation of smart grid: Risk-limiting dispatch. *Proceedings of the IEEE*, 99(1):40–57. 2011.

[105] H. Wang and X. Guo. An adaptive solution for Web service composition. In: *Proceedings of the 6th World Congress on Services, SERVICES-1 (Miami, FL, July 2010)*, pages 503–510. 2010.

[106] H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, and A. Bouguettaya. Adaptive service composition based on reinforcement learning. In: *Service-Oriented Computing*, volume 6470 of *Lecture Notes in Computer Science*, pages 92–107. Springer Berlin Heidelberg. 2010.

[107] C.C. White III. A survey of solution techniques for the Partially Observed Markov Decision Process. *Annals of Operations Research*, 32:215–230. 1991.

[108] C. Wu, K. Chen, C. Huang, and C. Lei. An Empirical Evaluation of VoIP Play-out Buffer Dimensioning in Skype. In: *Proceedings of the 19th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV, (Williamsburg, VA, June 2009)*. 2009.

[109] Y. Wu, C. Williamson, and J. Luo. On processor sharing and its applications to cellular data network provisioning. *Performance Evaluation*, 64(9–12):892–908. 2007.

[110] J. Young and X.Y. Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer-Verlag. 1999.

[111] A. Yousefi and D.G. Down. Request Replication: An alternative to QoS aware service selection. In: *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications, SOCA (OC Irvine, CA, December 2011)*, pages 1–4. 2011.

[112] T. Yu, Y. Zhang, and K.J. Lin. Efficient algorithms for Web services selection with end-to-end QoS constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6. 2007.

[113] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327. 2004.

[114] L. Zeng, C. Lingenfelder, H. Lei, and H. Chang. Event-driven quality of service prediction. *Proceedings of the 6th International Conference on Service Oriented Computing, ICSOC (Sydney, Australia, December 2008)*, pages 147–161. 2008.

[115] L. Zhang and H. Fu. Dynamic bandwidth allocation and buffer dimensioning for supporting video-on-demand services in virtual private networks. *Computer Communications*, 23(14–15):1410 – 1424. 2000.

[116] N.L. Zhang and W. Liu. Region-based approximations for planning in stochastic domains. In: *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence, UAI (Providence, RI, August 1997)*, pages 472–480. 1997.

[117] H. Zheng, J. Yang, W. Zhao, and A. Bouguettaya. QoS analysis for Web service compositions based on probabilistic QoS. In: G. Kappel, Z. Maamar, and H.R. Motahari-Nezhad, editors, *Service-Oriented Computing*, volume 7084 of *Lecture Notes in Computer Science*, pages 47–61. Springer Berlin Heidelberg. 2011.

[118] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya. QoS analysis for Web service composition. In: *Proceedings of the IEEE International Conference on Services Computing, SCC '09 (Bangalore, India, September 2009)*, pages 235–242. 2009.

[119] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya. QoS analysis for Web service composition. In: *Proceedings of the 2009 IEEE International Conference on Services Computing, ICSOC (Stockholm, Sweden, June 2009)*, pages 235–242. IEEE. 2009.

[120] M. Živković, J.W. Bosman, J.L. van den Berg, and R.D. van der Mei. Profit maximization with dynamic service selection in SOA. Submitted for publication.

[121] M. Živković, J.W. Bosman, J.L. van den Berg, R.D. van der Mei, H.B. Meeuwissen, and R. Núñez-Queija. Dynamic profit optimization of composite web services with SLAs. In: *Proceedings of the IEEE Global Telecommunications Conference, GlobeCom (Houston, TX, December 2011)*, pages 1–6. 2011.

[122] M. Živković, J.W. Bosman, J.L. van den Berg, R.D. van der Mei, H.B. Meeuwissen, and R. Núñez-Queija. Run-time revenue maximization for composite web services with response time commitments. In: *Proceedings of the IEEE 26th International Conference on Advanced Information Networking and Applications, AINA (Fukuoka, Japan, March 2012)*, pages 589–596. IEEE. 2012.