TECHNISCHE UNIVERSITÄT DARMSTADT

PERFORMANCE EVALUATION OF TRANSITION-BASED SYSTEMS WITH APPLICATIONS TO COMMUNICATION NETWORKS

vom Fachbereich Elektrotechnik und Informationstechnik der
TECHNISCHE UNIVERSITÄT DARMSTADT

zur Erlangung des Grades
Doktor rerum naturalium (Dr. rer. nat.)
Dissertation

von

SOUNAK KAR, M.STAT.,

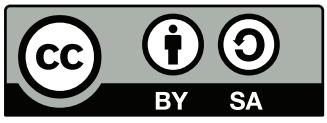
geboren am
28.01.1987 in Jalpaiguri, Indien.

Erstgutachter: Prof. Dr. -Ing. Ralf Steinmetz
Zweitgutachter: Prof. Dr. -Ing. Amr Rizk

Darmstadt 2021

The work of Sounak Kar was supported by the German Research Foundation (DFG) in the Collaborative Research Centre (SFB) 1053 "MAKI - Multi-Mechanism-Adaptation for the Future Internet" at the Technische Universität Darmstadt, Germany. The computation facility at Lichtenberg High Performance Computer at TU Darmstadt is also gratefully acknowledged.

Kar, Sounak: *Performance Evaluation of Transition-based Systems with Applications to Communication Networks*
Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2022
Tag der mündlichen Prüfung: 05.10.2021



To Pataka, the newest member of the family.

ABSTRACT

Since the beginning of the twenty-first century, communication systems have witnessed a revolution in terms of their hardware capabilities. This transformation has enabled modern networks to stand up to the diversity and the scale of the requirements of the applications that they support. Compared to their predecessors that primarily consisted of a handful of homogeneous devices communicating via a single communication technology, today's networks connect myriads of systems that are intrinsically different in their functioning and purpose. In addition, many of these devices communicate via different technologies or a combination of them at a time. All these developments, coupled with the geographical disparity of the physical infrastructure, give rise to network environments that are inherently dynamic and unpredictable. To cope with heterogeneous environments and the growing demands, network units have taken a leap from the paradigm of static functioning to that of adaptivity.

In this thesis, we refer to adaptive network units as transition-based systems (TBSs) and the act of adapting is termed as transition. We note that TBSs not only reside in diverse environment conditions, their need to adapt also arises following different phenomena. Such phenomena are referred to as triggers and they can occur at different time scales. We additionally observe that the nature of a transition is dictated by the specified performance objective of the relevant TBS and we seek to build an analytical framework that helps us derive a policy for performance optimization.

As the state of the art lacks a unified approach to modelling the diverse functioning of the TBSs and their varied performance objectives, we first propose a general framework based on the theory of Markov Decision Processes. This framework facilitates optimal policy derivation in TBSs in a principled manner. In addition, we note the importance of bespoke analyses in specific classes of TBSs where the general formulation leads to a high-dimensional optimization problem. Specifically, we consider performance optimization in open systems employing parallelism and closed systems exploiting the benefits of service batching. In these examples, we resort to approximation techniques such as a mean-field limit for the state evolution whenever the underlying TBS deals with a large number of entities. Our formulation enables calculation of optimal policies and provides tangible alternatives to existing frameworks for Quality of Service evaluation. Compared to the state of the art, the derived policies facilitate transitions in Communication Systems that yield superior performance as shown through extensive evaluations in this thesis.

ZUSAMMENFASSUNG

Seit Beginn des einundzwanzigsten Jahrhunderts haben Kommunikationssysteme einen fundamentalen Wandel erlebt. Dieser Wandel hat moderne Kommunikationsnetze in die Lage versetzt, eine sehr große Vielfalt und einen großen Umfang der Anforderungen der Anwendungen zu unterstützen. Im Vergleich zu herkömmlichen Kommunikationssystemen, die über eine einzige Kommunikationstechnologie kommunizierten, verbinden heutige Netzwerke eine sehr hohe Anzahl an Systemen, die mit der Fähigkeit ausgestattet sind, gleichzeitig über verschiedene Technologien zu kommunizieren. All diese Entwicklungen, gepaart mit der geografischen Disparität der physischen Infrastruktur, führen zu Netzumgebungen, die von Natur aus dynamisch und unvorhersehbar sind. Um mit solchen Herausforderungen und gleichzeitig mit den wachsenden Anforderungen fertig zu werden, haben Netzkomponenten einen Paradigmenwechsel von der statischen Funktionsweise zu dem der Adaptivität, bzw. der transitionsbasierten Systeme, gemacht.

In dieser Arbeit bezeichnen wir adaptive Netzkomponenten als transitionsbasierte Systeme (TBS), und der Akt der Anpassung wird als Transition bezeichnet. Wir stellen fest, dass sich TBS nicht nur in unterschiedlichen Umgebungsbedingungen befinden, sondern dass ihr Anpassungsbedarf auch nach verschiedenen Ereignissen entsteht. Solche Ereignissen werden als Auslöser (Trigger) bezeichnet und sie können auf unterschiedlichen Zeitskalen auftreten. Weiterhin wird die Art einer Transition durch das spezifizierte Leistungsziel des betreffenden TBS diktiert. In dieser Arbeit wird ein analytischer Rahmen geschaffen um Strategien zur Leistungsoptimierung unter Adaptivitätsaspekten in TBS abgeleitet.

Da dem Stand der Technik ein einheitlicher Ansatz zur Modellierung der unterschiedlichen Funktionsweisen der TBS und ihrer verschiedenen Leistungsziele fehlt, schlagen wir zunächst einen allgemeinen analytischen Rahmen vor, der auf der Theorie der Markov-Entscheidungsprozesse basiert. Dies erleichtert die fundierte Ableitung optimaler Strategien in TBS. Darüber hinaus zeigen wir die Bedeutung von systemspezifischen Analysen in bestimmten Klassen von TBS, bei denen die obige Formulierung zu einem hochdimensionalen Optimierungsproblem führt. Konkret betrachten wir die Leistungsoptimierung in sowohl offenen Systemen, die Parallelität nutzen, als auch in geschlossenen Systemen, die Batching-Mechanismen verwenden. In diesen Systembeispielen greifen wir auf Approximationstechniken zurück, wie z. B. eine Mean-Field-Analyse, wenn das zugrundeliegende TBS aus einer großen Anzahl von Komponenten besteht. Unsere Analyse führt zu einer kompakten Bewertung von Dienstgütemetriken

für adaptive Kommunikationssysteme in geschlossener Form und darüber hinaus zur Berechnung optimaler Transitionsstrategien für offene und geschlossene transitionsbasierte Kommunikationssysteme.

PREVIOUSLY PUBLISHED MATERIAL

This thesis builds on material that has been already published in various journals and peer-reviewed conferences. The following publications are included here in parts or in an extended version:

- B. Alt, M. Weckesser, et al. (2019). "Transitions: A Protocol-Independent View of the Future Internet." In: *Proceedings of the IEEE* 107.4, pp. 835–846.

- S. Kar, A. Rizk, et al. (2018). "Multi-interface Communication: Interface Selection Under Statistical Performance Constraints." In: *2018 30th International Teletraffic Congress (ITC 30)*. Vol. 02, pp. 7–12.

- W. R. KhudaBukhsh, S. Kar, et al. (2020a). "Generalized Cost-Based Job Scheduling in Very Large Heterogeneous Cluster Systems." In: *IEEE Transactions on Parallel and Distributed Systems* 31.11, pp. 2594–2604.

- S. Kar, R. Rehrmann, et al. (2020). "On the Throughput Optimization in Large-scale Batch-processing Systems." In: *Performance Evaluation* 144, pp. 102–142.

All publications mentioned in this section are a result of joint effort of the respective co-authors and were primarily facilitated by the DFG Collaborative Research Center MAKI[1]. MAKI brings together professionals from the areas of Computer Science, Electrical Engineering, and Economics to identify major hurdles for a future-ready internet and propose respective solutions. As a natural consequence, the collaborative efforts from different disciplines are visible in the publications. However, the author's contribution has been primarily, although not exclusively, focused on formulating necessary abstraction of a real-world problem and evaluating such frameworks through simulations. To present the contributions with due consistency, notations, algorithms, figures and experimental evaluations have been borrowed from respective publications. Nonetheless, the overall content has been adapted according to the relevance of the published material to this thesis. The table below summarizes the applicability of the publications in the context of the specific chapters. The contributions from the author of this thesis in the context of the *relevant* publications are also delineated in the next paragraphs.

1 MAKI– Multi Mechanism Adaptation for the Future Internet. `https://www.maki.tu-darmstadt.de/` [Accessed: June 2021].

Publication	Chapters
Alt, Weckesser, et al. 2019	Chap. 1
Kar, Rizk, et al. 2018	Chap. 3
KhudaBukhsh, Kar, et al. 2020a	Chap. 4
Kar, Rehrmann, et al. 2020	Chap. 5

Introduction of the thesis (Chap. 1) presents a framework for analyzing performance of TBSs, which largely draws upon elements from the paper (Alt, Weckesser, et al. 2019). The main building blocks of this paper are in turn borrowed from the literature on Dynamic Software Product Lines (DSPL) and Markov Decision Processes (MDPs). In this paper, the responsibility of the author of this thesis was to lay out the relevant concepts of MDPs required for building the framework; see (**Sect. II-B**; Alt, Weckesser, et al. 2019).

Chap. 3 of this thesis proposes a periodic interface selection strategy in a parallel open TBS which led to the publication (Kar, Rizk, et al. 2018). In the context of this publication, the co-authors formulated the problem and contributed to the paper structure and presentation. Author of this thesis was responsible for the analytical derivations, i.e., proposing the waiting time bound-based selection strategy and the calculation of the service envelope for Markov-modulated processes. Furthermore, the numerical evaluations of the proposed scheme under different network conditions were also carried out by him.

In Chap. 4, we delve into the topic of performance optimization in another open parallel system, where transitions are triggered by certain events. This chapter is broadly based on the publication (KhudaBukhsh, Kar, et al. 2020a). The paper seeks to address the problem of scheduling in large clusters with finite buffers and the first author of the paper introduced the concept of a cost-based scheduling to address the problem and made subsequent analytical contributions. While the fourth author identified the relevance of the problem and led a concrete formulation, all authors contributed with writing and presentation of the paper. The author of this thesis was primarily responsible for the numerical section of this paper. This includes the numerical illustration of the following: comparison of different cost-based scheduling strategies, extent of load-balancing achieved via different cost functions, performance implication of different strategies in terms of server backlog, and the effect buffer size on server blocking.

In Chap. 5, performance optimization of a closed batching TBS is discussed. This chapter is based on the publication (Kar, Rehrmann, et al. 2020), where the underlying throughput optimization problem in closed batching systems was identified by the last author of the paper. While the straightforward CTMC-based optimization was done by the author of this thesis, the third author proposed the mean-field formulation to solve the optimization problem scalably and derived the analytical results for the single job-type case (**Sect. 4**; Kar, Rehrmann, et al.

2020). The analytical derivations for the two job-types case with differential batch sizes and the multiple job-types case with same batch size were done by the author of this thesis. The author of this thesis was also responsible for running the simulations experiments in MATLAB and analyzing the data from real-world experiments, run in a prototype of a commercial database by the second author of the paper. The organization and presentation of the paper were shaped by extensive contributions from the all coauthors.

Finally, Chap. 6 focuses on the aspect of overall networked system quality. Specifically, the problem of Active Queue Management is addressed in this chapter in a principled manner. The author of the thesis was responsible for proposing the Semi-Markov Decision Process-based formulation of the decision problem for both negligible and non-negligible RTT case (both single and multi-flow scenarios). The experiments showing the efficacy of the proposed AQM algorithm over the state of the art in simulation were run by him as well. It goes without saying that the publications and the contributions in this thesis in general came to fruition due to the relentless support the author received from both his supervisors, Prof. Steinmetz and Prof. Rizk, especially in terms of problem identification, solution ideation, organization, and presentation.

Apart from the publications mentioned above, the following papers were part of the author's doctoral research, although not covered in this thesis. The topics of these publications are outside the scope of the material covered here:

- S. Kar, R. Hark, et al. (2018). "Towards Optimal Placement of Monitoring Units in Time-Varying Networks Under Centralized Control." In: *Measurement, Modelling and Evaluation of Computing Systems*. Cham: Springer International Publishing, pp. 99–112.

- S. Kar and A. Rizk (2020). "A Delicate Union of Batching and Parallelization Models in Distributed Computing and Communication." In: *2020 IFIP Networking Conference (Networking)*, pp. 534–538.

- W. R. KhudaBukhsh, S. Kar, et al. (Mar. 2019). "Provisioning and Performance Evaluation of Parallel Systems with Output Synchronization." In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 4.1.

- W. R. KhudaBukhsh, B. Alt, et al. (2018). "Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies." In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 1–9.

- R. Hark, M. Ghanmi, et al. (2018). "Representative Measurement Point Selection to Monitor Software-defined Networks." In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 511–518.

ACKNOWLEDGMENTS

My four-year stint in Darmstadt is coming to an end. As I look back, I tend to realize that it would not have been the same without the help and assistance I received here. First of all, I would like to thank Amr for his continued support and attention. I am equally grateful to Prof. Steinmetz for extending me the opportunity to complete my doctoral studies here at KOM. The list would be incomplete if I did not acknowledge the counsel and guidance of my collaborators. It has been a rewarding experience to work with Arpan Mukhopadhyay, Florin Ciucu, and Heinz Koeppl. I further thank Prof. Klein and Prof. Steinke for agreeing to be on the thesis panel.

I also owe it to the advice, support, and company of many of my colleagues here: the ever-helpful admin team at KOM- Frau Scholz-Schmidt, Frank, and Karola, my ACS counterparts- Yassin, Christian, Manisha, Rhaban, Sonja, and Patrick, the KN1 team- Wael, Anna, and Jannis, and the BCS folks- Anam and Bastian. My Darmstadt days would not have been as enjoyable without their regular presence. Further, I am grateful to Tobi, Christoph, Julian, Ralf, Pratyush, and the team for preparing a wonderful Doktorwagen. I was delighted to have such a nice ride on the big day and really appreciate their effort and craftsmanship. Coming back to my time at Darmstadt, I am certainly going to miss the weekend dinners with the IMP folks, hosted by Uttam.

I would also take this opportunity to expressly thank my ISI friends, Wasiur and Tulasi, who have been inspirational for my return to academia. The long Skype calls with them on research roadblocks and things outside of work helped me stride through the stages of my doctoral studies. Finally, I want to thank my family for being the way they are- it is comforting to know that I have people by my side no matter what.

CONTENTS

LIST OF ABBREVIATIONS

AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
CBS	Cost-Based Scheduling
CCDF	Complementary Cumulative Distribution Function
CLT	Central Limit Theorem
CTMC	Continuous Time Markov Chain
FCFS	First Come First Serve
HMM	Hidden Markov Model
JSQ	Join the Shortest Queue
LLN	Law of Large Numbers
MCMC	Markov Chain Monte Carlo
MDP	Markov Decision Processes
MMOO	Markov Modulated On Off
ODE	Ordinary Differential Equation
QoS	Quality of Service
RTT	Round Trip Time
SMDP	Semi Markov Decision Processes
TBS	Transition-Based System
TCP	Transmission Control Protocol

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
$\mathbb{N}$	The set of natural numbers.
$[N]$	The set $\{1, 2, 3, \ldots, N\}$ for $N \in \mathbb{N}$.
$\mathbb{Z}$	The set of integers.
$\mathbb{Z}_+$	The set of nonnegative integers.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}_+$	The set of positive real numbers.
S^n	The n-fold cartesian product of a set S.
$\mathcal{B}(R)$	The σ-field of Borel subsets of $R \subseteq \mathbb{R}^n$, $n \geq 1$.
$\mid A \mid$	The cardinality of a set A.
$\mathbb{1}_A$	The indicator (or characteristic) function of a set A.
$\mathsf{P}(E)$	Probability of an event E.
$\mathsf{E}[X]$	Expectation of a random variable X.
$\stackrel{\mathcal{D}}{\Longrightarrow}$	Weak convergence.
$O()$	$f(n) = O(g(n))$ iff $\lim_{n\to\infty} f(n)/g(n) = c \in \mathbb{R}$.
$o()$	$f(n) = o(g(n))$ iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\omega()$	$f(n) = \omega(g(n))$ iff $g(n) = o(f(n))$.

1

INTRODUCTION

'Data is the new oil'. While this recent cliché has elicited different responses in different quarters over its consequences, the phrase itself has largely remained uncontested. And over the last decade, our expertise to gather and process data has only multiplied. While we amaze at the ability of emerging technologies to process huge amounts of data and derive meaningful insights, our capability to gather this information remains less appreciated. Digital information gathering in the recent past has been supported by the proliferation of communication networks and the origin of this phenomenon can be traced back to our ability to connect computers.

Early computer networks mostly belonged to standalone organizations and exchanged information primarily via a single technology, namely copper wire. These networks were not only limited in their size and the choice of communication technology, but also in terms of the functions carried out for a secure and orderly communication. While the basic functionalities, such as packet forwarding, routing, and access control, remain the same, their flexibility and complexity have taken a giant leap. This has been necessitated by the enormous growth of modern communication networks and the diversity of the available communication technologies. Today, the internet connects 10 billion devices [1] and many of them can communicate with each other via a plethora of technologies– WiFi, LTE and Bluetooth to name a few. In addition, modern devices are capable of switching between communication technologies at runtime or even using them together (Barré, Paasch, et al. 2011).

Apart from their size and diversity, modern networks are far more dynamic in nature than their early counterparts. This evolution can be attributed to both the change in the physical nature of the network, such as the addition of mobile phones or, more recently, internet balloons (Katikala 2014) and the growing user demands, such as security and resilience. The crucial aspect of Quality of Service (QoS) has also contributed to this change. To maintain a certain level of QoS, networks often follow topology control algorithms to cope with the changing network environment, such as link failure and the varying degree of available resources in the connecting components. Such scenarios are becoming increasingly frequent and are causing networks to move away from bespoke

1 Are Enterprises Ready for Billions of Devices to Join the Internet? Article by D. Puglia, FrontRange. `https://www.wired.com/insights/2014/12/enterprises-billions-of-devices-internet/` [Accessed: June 2021].

topology control algorithms to a more universal approach (Stein, Frömmgen, et al. 2016).

The components of modern networks themselves are no less dynamic or diverse and this makes meeting QoS requirements even more challenging. To take an example, many of today's networks connect distributed complex event processing systems. In such a system, a single operator placement algorithm often falls short of fulfilling user-defined performance demands under dynamic environments, which calls for adaptive placement algorithms (Luthra, Koldehofe, et al. 2021).

Needless to say, meeting growing user demands under changing environment requires seamless execution of basic network functionalities like routing and access control. The scale of present-day networks poses new challenges even to such basic network management; although the difficulties have been alleviated by contemporary hardware capabilities. For example, programmable switches are far more flexible than their predecessors and are often able to maintain a desired level of performance by adapting to changing network state (Bleuler, Laumanns, et al. 2003; Kundel, Blendin, et al. 2018).

Goal of the thesis: In this thesis, we build a mathematical abstraction of the dynamics of adaptive systems, which helps us frame a policy for performance optimization. To that end, we first adopt a universal framework that is capable of accounting for the diversity of the dynamics and the performance goals of adaptive systems. Subsequently, the framework provides a general tool for QoS optimization with respect to desired performance metrics. Compared to the state of the art that focuses on specific abstractions, our framework provides a general tool for analyzing adaptive network systems. While we explain the flexibility of the framework, we also note that the generality comes at a cost of increased computational burden. Thus, we turn to bespoke analysis whenever deemed pragmatic. The advantages of our approach in such specific cases are also explained while comparing them with the existing literature. Next, we take an example of a typical present-day network connecting adaptive systems and illustrate their performance goals. Through this example, we formulate the research questions this thesis aims to address and explain their relevance as well.

1.1 A COMMUNICATION NETWORK SCENARIO

Consider the communication network in Fig. 1.1 connecting systems commonly seen in today's networks. The cloud-based infrastructure comprising network switches forms the core of the network. On the left hand side, we have a set of users connecting to a data processing system that serves jobs in a batch of an arbitrary yet fixed size. Job batching is a common technique to amortize operational overhead over multiple jobs to reduce per-job service time. An

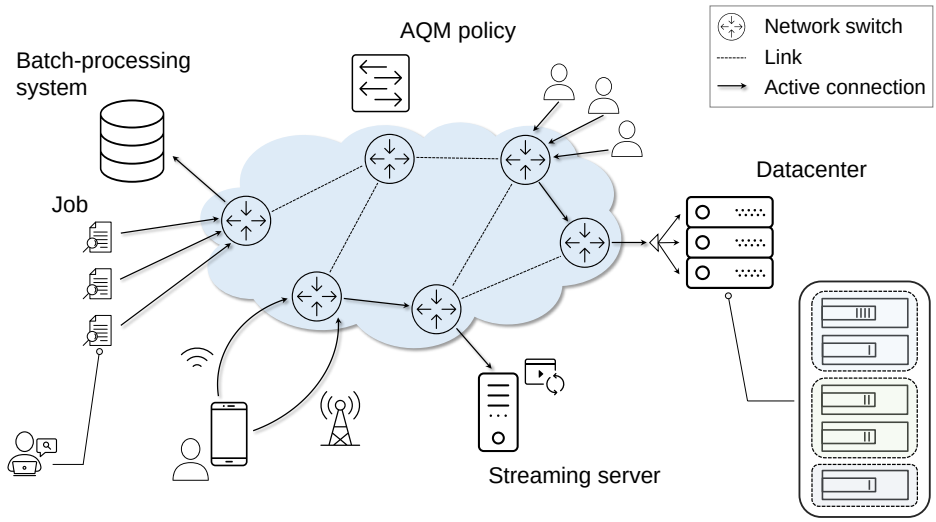


Figure 1.1: A communication network connecting transition-based systems. On the left, users submit jobs to a batch-processing system that serves a predetermined number of users. The system aims to maximize its throughput by serving jobs in batches. The optimal batch size is recalculated whenever there is a transition, i.e., a system reconfiguration. At the bottom, we have a user fetching content from a streaming server. The user seeks to minimize the latency over the used communication interface by alternating between them periodically. On the right, we show a datacenter comprising heterogeneous clusters that might have a hybrid user-defined objective, e.g., uniform load distribution when the backlog is small and delay minimization when the workload is high. In the core of the network, we see an AQM-capable switch, which aims to optimize its throughput while keeping the delay under a strict threshold. To meet its objective, the switch devises a policy that drops or admits an incoming packet by looking at its current backlog and the arrival rates from adjoining switches.

example of a batch processing system could be a database or a certain type of network card in Linux-based systems. The objective of the system here is to maximize its expected throughput, which it does by choosing an appropriate batch size. Although increasing the batch size leads to a faster per job service time on average, the problem becomes non-trivial when the system has multiple cores to serve the jobs. This is because increasing the batch size beyond a limit causes the core(s) to idle.

At the bottom of the figure, we see that a user is connected to a streaming server via the network core. The connection to the core can be established via multiple interfaces. The user device periodically selects one of the available interfaces at runtime to meet its goal of minimizing packet delay over the used

interface. A typical example of such a scenario is a mobile user fetching video content from a server where the connection to the server can be established via WiFi or LTE. Evidently, the choice of the interface for the next period should take into account the strength of the signal over interfaces and their existing backlogs. Further, variability and the lack of perfect knowledge of signal strengths can be supplemented by drawing inference from past observations.

The network also contains a datacenter made of heterogeneous clusters shown on the right hand side of Fig. 1.1. These clusters containing homogeneous servers might represent different geographical locations or the suitability of the servers for different jobs, for example. The datacenter has a central scheduler that assigns an incoming job to one of the servers according to the current workload and the suitability of the cluster with regards to the said job. To give an example, a machine learning job might have high resource requirement that can only be met by certain clusters. A user-defined performance objective is fed into the scheduler via an encoded cost function.

We look at the consequent evolution of the workload distribution in each cluster over a specified time interval. This metric lets us compare the effectiveness of different cost functions in terms of fulfilling a predefined goal.

In the core of the network, we see a switch deploying Active Queue Management (AQM). AQM is an algorithm that drops an incoming packet even before the buffer of the switch becomes full. The drop pattern follows a certain rule called AQM policy. The objective of AQM is to counter delays in sizeable buffers due to the queueing of a large number of packets, without sacrificing too much throughput. Put another way, the switch aims to maximize its throughput while adhering to a user-specified delay guarantee. To that end, it takes the current backlog and the packet arrival rates of the existing flows as inputs and decides whether to drop or admit the incoming packet.

A common feature of the systems described above is that they have a well-defined metric of performance and are transition-based. That is, as the systems are reconfigured or their state changes, the functioning adapts in order to achieve optimal performance with respect to the predefined metric. Further, the functioning involves making certain changes in parameter(s) or variable(s) relevant to the state of the system. We call these parameters and variables control parameters and control variables, respectively. For example, as the batch-processing system on the left hand side of Fig. 1.1 is upgraded, a new optimal batch size k^* is calculated and the system begins to serve jobs in batches of size k^* to maximize its expected throughput. Thus, the batch size k serves as the control parameter of this system. Unlike the batch-processing system where the control parameter is changed in a static manner, the control variable is changed dynamically in the AQM-enabled switch. We see that the packet arrival rates are the control variables in this context and we change them one at a time by dropping or admitting an

incoming packet. It is intuitive that the pattern of dropping packets should look quite different when the current incoming rate and the existing backlog are high as against when the buffer is relatively empty. Therefore, the state evolution and corresponding remedial action to optimize performance can have varying degree of complexity depending upon the underlying system.

To this end, we first propose a general framework that builds on the existing methodologies of Dynamic Software Product Lines (DSPL) and Markov Decision Processes (MDP). We elaborate on this approach in Sect. 1.2. We further see that transitions are triggered in the underlying systems by different phenomena that occur at diverse time scales. For example, the functioning of the batch-processing system is adapted only upon system reconfiguration, whereas the AQM-enabled switch takes a decision whenever a packet arrives.

We first explore the effect of transitions on the performance of parallel systems as formulated in the following research question:

***Research Question 1**: What is the impact of time-triggered and event-triggered transitions on the Quality of Service in parallel communication and computing systems?*

We address this question specifically in the context of multi-interface communication systems in Chap. 3. Here the system adapts at the beginning of every period having fixed time length and we propose a policy to minimize delays in the system. In contrast, transitions in the parallel system considered in Chap. 4 are triggered by certain events and we investigate their effect on a general user-defined performance metric.

Next, we focus on the performance evaluation of closed batching systems where the batch size is changed upon system reconfiguration. This leads us to the following research question:

***Research Question 2**: How to optimize a given Quality of Service for closed transition-based batching systems?*

Noting that reconfiguration can lead to arbitrary system sizes, we produce an exact analysis as well as a scaling approximation for the performance of transition based batching systems in Chap. 5. While the former approach yields results with higher accuracy, the latter is more suited for large systems due to its efficient computation.

Having analyzed specific network systems, we also focus on the aspect of overall network performance. We see that the complex functioning of the systems connected to the network is supported by basic network activities like packet forwarding and routing, which motivates us to consider the following research question:

***Research Question 3**: How to find optimal transition policies for network elements to maximize the overall networked system quality?*

To answer this question, we analyze the performance of an AQM-enabled switch in Chap. 6. AQM-enabled switches aim to maximize flow throughput while keeping the delay under a threshold. We propose a policy that prescribes whether to drop/admit an incoming data packet by looking at the state of the switch. Instead of focusing on the immediate gain, our policy aims to maximize the performance over a long time horizon. Next, we introduce the general framework that helps us describe the dynamics of adaptive network systems.

1.2 BACKGROUND: TRANSITION-BASED SYSTEMS

In this section, we focus on formulating the necessary abstractions to describe adaptive systems. As mentioned earlier, a common property of the systems we consider is that each of them has a predefined performance metric. We optimize system performance with respect to this metric by making certain changes in the control variable(s) or the control parameter(s) of the system. These changes can be executed either dynamically at runtime or in a static fashion as the system is (re)configured. Note that a control variable or a parameter is part of a broader conceptual element, called mechanism. A mechanism is defined as a set of coordinated functional units specific to a realization of the considered system (Frömmgen, Hassan, et al. 2016). To give an example, a mechanism in the context of the datacenter on the right of Fig. 1.1 might refer to a scheduling policy. The changes in a mechanism cause the state of the system to update and this phenomenon is termed as transition, which explains why the systems are called transition-based. In our context, transitions are followed by a trigger, which might refer to an event or a fixed time-interval. For example, in the context of the mobile user shown at the bottom of Fig. 1.1, a transition occurs at the beginning of every period, which we also term as decision epoch. On the contrary, in the datacenter example, a transition might be necessary once the workload distribution of the servers moves to a different region where the regions are specified by the cost function of the scheduler.

In this section, we lay down a framework that is general enough to handle the evolution and performance optimization in transition-based systems. We note that it is imperative to choose a minimal sufficient level of granularity such that the framework does not suffer from the curse of dimensionality while remaining adequate. With that vision, we borrow the oft-used methodologies, DSPL and MDP, to build our framework. The first component of this framework helps us identify the minimal set of features required to form an adequate representation of the system where a feature might refer to a mechanism or its component, which might be a variable or a parameter of the mechanism. Further, a feature might be system-specific or might represent the environment in which the system resides. We say that the first type of feature describes system configurations

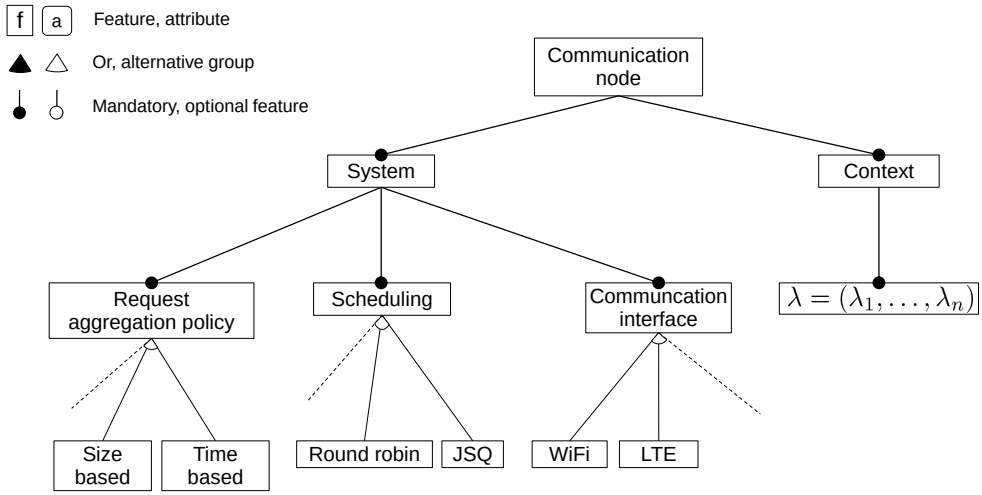


Figure 1.2: A context feature diagram of the communication node connected to a streaming server shown at the bottom of Fig. 1.1. The context includes the parameters for the channel condition of each communication interface. The system configuration has the mandatory system features, communication technology, the scheduling algorithm and the request aggregation policy. Each feature can be exclusively in one state as shown by respective children features forming an alternative group. At every decision epoch, the system actively selects or deselects one of the children features to fulfill its performance objective.

where the second type represents context. Having identified the key features and their corresponding domain, we use MDP to model the state evolution and the process of performance optimization.

A key concept of the DSPL methodology is to use a feature diagram to represent the configuration space of a transition-based system that details the domain of each feature and their dependencies (Kang, Cohen, et al. 1990). The features are usually organized using a tree structure where the decomposition relationship is shown by parent-children hierarchy. The decomposition relationship can be one of four types: mandatory, optional, or and alternative. Further, functional relationships between features are represented by a cross-tree structure and can be of the type require, exclude or might even be formulated as a propositional expression. Finally, non-functional characteristics of a feature are captured by feature attributes, which have their corresponding domains. Thus, the feature space and its constraints can be precisely represented using a feature diagram (Weckesser, Kluge, et al. 2018a) as we show in Fig. 1.2 through an example.

Equipped with the feature space, we look for a tool that enables us to derive the optimal decisions for dynamic systems and MDP comes as a natural choice

due to its flexibility. An MDP is defined as a tuple $< \mathcal{X}, \mathcal{A}, \mathsf{P}, R >$. Here $\mathcal{X}$ denotes the state space, $\mathcal{A}$ is the action space and P and R denote the transition probability function and the reward function given a certain action is taken in a particular state, respectively. We emphasize that the state space of the MDP is indeed the feature space in DSPL terminology.

Let S and ξ respectively denote the tuple of system and context features that adequately describe the underlying system. Further, let $\mathcal{S}$ denote the set of possible values S can assume and Ξ denote the context space. Put another way, $\mathcal{S}$ is the set of system configurations and Ξ denotes the set of possible contexts. Then, the state space $\mathcal{X}$, also called the feature space of the system, can be expressed as $\mathcal{X} = \mathcal{S} \times \Xi$. To give an example, let us consider the mobile user fetching content from the video-streaming server via WiFi and LTE. We see that the context is given by the features denoting the signal quality over these interfaces. Each of these features has an attribute that parametrizes the signal quality. For example, if we assume that the time to send a packet of unit size over an interface takes an exponentially distributed time, the (service) rate of that exponential distribution might act as a corresponding attribute. Further, we see that the system state is adequately represented by the backlogs at the two different interfaces. Thus, the backlogs and the service rates at the two interfaces are the system and the context features of the model, respectively. To obtain the state space for the corresponding MDP formulation, we first assume that the buffers at each of these interfaces are arbitrarily large. Then, the set of system configurations are given by $\mathcal{S} = \mathbb{Z}_+^2$. Further, the service rates can be any positive quantity, which lets us write the context space as $\Xi = \mathbb{R}_+^2$. Therefore, the feature space or the state space can be expressed as $\mathcal{X} = \mathbb{Z}_+^2 \times \mathbb{R}_+^2$ in this case.

The set $\mathcal{A}$ in an MDP denotes the set of available actions, which in our context refers to the changes that can possibly be executed on the control variable(s) or the parameter(s). For example, in the context of the switch deploying an AQM policy, the action that can be taken on the arrival of a packet is either to drop or admit it. Since the action in this case is a binary variable we can encode the action space as $\mathcal{A} = \{0, 1\}$. Same goes for the mobile user as every decision epoch, it can only choose one of the two interfaces, namely WiFi or LTE.

The MDP framework adequately models transition-based systems in both discrete and continuous time. Under this framework, the systems make control decisions, i.e., transitions only at discrete time points. Denoting the state of the system at the t-th transition/decision epoch as $\mathsf{X}_t = (\mathsf{S}_t, \xi_t)$ and the corresponding action as Λ_t, we see that $\mathsf{P}\left(\mathsf{X}_{t+1} | \mathsf{X}_t, \Lambda_t\right)$ denotes the probability of the event that the system state during the next transition is X_{t+1}. For all transition-based systems considered here, P is time-homogeneous. To give an example, in the context of the AQM-capable switch serving a single flow where we represent the state as a tuple of backlog at the switch and the packet arrival rate, the

transition probability ascribes positive mass at all possible transitions between two consecutive arrivals. Here, given the action, the change in queue length and the arrival rate are independent and state transition probability can be expressed as a product of coordinate-wise transition probabilities. Further the arrival rate changes deterministically following a drop or admit action and hence the corresponding transition probability can only be 0 or 1. The transition probabilities for queue lengths can also be suitably derived assuming certain interarrival and service time distribution and the final state transition probabilities can be duly calculated.

The quantity $R(\mathsf{X}_t, \Lambda_t)$ in an MDP expresses the immediate gain during the t-th transition in terms of the performance metric of the underlying system. In the context of the AQM-enabled switch, we seek to maximize the throughput subject to a hard constraint on the observed packet delay. This objective is encoded into the reward function R, which is a linear combination of a prohibitive penalizing factor when the delay guarantee is breached and a function of the change in the throughput following a drop/admit action. Equipped with the reward function R and the transition probabilities P, one can derive the optimal policy π^* from the set of policies $\Pi = \{\pi | \pi : \mathcal{X} \mapsto \mathcal{A}\}$ in terms of average long term accumulated reward or the total long term discounted reward, depending upon the performance objective of the system. The corresponding derivation process usually follows standard algorithms such as value iteration or function approximation (Sutton and Barto 2018). Note that $R(\mathsf{X}_t, \Lambda_t)$ in some cases denotes the cost of a transition and the policy derivation algorithm in such cases naturally seeks to minimize the accumulated cost. For a thorough treatment of the underlying algorithms to determine the optimal policy, the reader is referred to (Sutton and Barto 2018).

Although the MDP framework serves as a general tool for modelling the behaviour and performance of transition-based systems, it often burdens the analysis with considerable computational cost. Note that an increase in the number of variables in the state representation leads to an exponential blow up of the size of the state space, which might render the existing policy derivation algorithms infeasible. Thus, in practice, it is sometimes pragmatic to trade away the generality of the MDP framework for more tractable ad-hoc models. For example, in the context of the AQM-enabled switch, the analysis of the dynamics requires the flexibility of the MDP framework and we use this general framework to derive the optimal policy. However, in the case of the batch-processing system, the control parameter for optimizing performance is updated only in a static fashion, for example, upon system reconfiguration. As we take a static approach to optimizing for the given metric of performance here, the associated computational cost of the MDP framework makes it untenable. Consequently, we perform a bespoke analysis of the system dynamics and look at its long-term behaviour in terms of the system parameters. This lets us calculate the optimal value of the

control parameter, i.e., the batch size k with respect to the desired metric, namely the expected throughput of the system. We recalculate the optimal value of k whenever the setup changes. Also, in our pursuit of optimizing performance, we sometimes resort to near-optimal policies or heuristics depending upon the individual case. Nevertheless, we demonstrate the efficacy of the suggested policies over the candidate policies for each system. Next, we summarize the overall contributions of this thesis and specify the chapters where a particular TBS is explored in greater detail.

1.3 CONTRIBUTIONS AND STRUCTURE OF THE THESIS

Contributions: In this thesis, we focus on the effects of transitions on the performance of TBSs. We first consider parallel systems where transitions can be either time-based or event-based. To that end, we propose a transition policy that improves system performance. We also consider closed batch-processing systems where the performance metric depends on the chosen batch size. We provide an exact method to calculate the optimal batch size and further propose an approximation for large systems. Finally, we observe that the overall performance of a network hinges on basic network activities such as packet forwarding and routing. Accordingly, we consider an AQM-enabled switch that aims to maximize its throughput while adhering to a certain delay constraint. We formulate an optimal policy that makes an informed decision by looking at the state of the switch and the relevant network environment.

Structure: In the rest of the thesis, we first discuss the existing approaches to performance optimization in respective systems and subsequently elaborate on our approach to each system as detailed below:

- In Chap. 2, we discuss the related work for performance optimization in transition-based systems. We include approaches for general transition-based systems as well as the specific solutions proposed in the literature in the context of the individual systems represented in Fig. 1.1.

- Chap. 3 elaborates on transitions in a parallel system, which periodically selects the most suitable path in order to optimize delay. Recall that an example of such a system, a mobile user fetching content from a video streaming server, is shown at the bottom of Fig. 1.1. For such systems, we propose a criterion to pick the most suitable path at the beginning of each period and demonstrate its efficacy vis-a-vis periodic variants of standard alternatives like round robin or join the shortest queue (JSQ) policy.

- In Chap. 4, we discuss performance optimization in another parallel system where transitions are triggered by certain events. In this context, we take

the example of a datacenter where the scheduling policy is based on a cost function that defines the transition-triggering events: the migration of server workloads from one interval to another. The objective of delay optimization is expressed via the metric of time evolution of workload distribution for each constituent cluster. Here, we adopt a mean-field approach to model the cluster occupancy where the size of the datacenter is assumed to be infinitely large, subject to the condition that each cluster contains a positive proportion of servers. We show that the time evolution suggested by the model agrees with the corresponding empirical behaviour.

- In Chap. 5, we discuss throughput optimization in a batch-processing system shown on the left hand side of Fig. 1.1. Recall that transitions in such systems happen only upon system reconfiguration. To model the system evolution, we first use continuous time Markov chain (CTMC) as a tool. Due to the infeasibility of the CTMC model for a large system size, we resort to mean-field techniques and derive the asymptotic (in terms of system size) long-term performance of the system. We show that the performance can be reduced to a simple expression involving system parameters, which evidently includes the control parameter- the batch size. We finally evaluate the model predictions and show that they agree with the real-world findings.

- In Chap. 6, we discuss transitions in the AQM-enabled switch. Transitions are more frequent here than in our previous example and the system analysis requires the generality of the MDP framework. The state of the switch changes at every arrival epoch and a drop/admit decision is made. We come up with a principled policy that takes the current buffer-filling and the arrival rates of incumbent flows as inputs to suggest a drop/admit action. We compare the performance of our policy numerically with the state of the art, which are either heuristic-based or require experimental parameter engineering.

- Finally, we summarize the contribution of the thesis in Chap. 7 where we also discuss possible avenues for future research.

2

RELATED WORK

The goal of this thesis is to provide a mathematical framework that enables performance analysis of adaptive systems and optimization thereof. To meet that objective, we first proposed a framework in Sect. 1.2, built using tools from Dynamic Software Product Lines (DSPL) and Markov Decision Processes (MDP). While the former facilitates an appropriate state space representation of the system, the latter enables us to model the system behaviour and subsequent performance optimization process. In the following, we discuss existing approaches for modelling adaptive/transition-based systems (TBSs) and primarily focus on the aspects of state space modelling, system dynamics and practical applications of such models.

In the DSPL literature, there are two distinct approaches for modelling state space: architectural models and feature models (Krupitzer, Roth, et al. 2015). While the popular Rainbow framework (Garlan, Cheng, et al. 2004) uses architectural models, feature models form the basis for the well-known FUSION framework (Elkhodary, Esfahani, et al. 2010). The latter, which is used for building self-adaptive software systems, uses an online learning-based approach to find the impact of transition decisions while allowing fine-tuning of the system parameters at runtime. A visible shortcoming of these example frameworks is that they only focus on system features and do not model the context features. In contrast, using the framework of performance-influence models, the work (Weckesser, Kluge, et al. 2018b) provides a solution for obtaining optimal reconfiguration decisions to achieve context-dependent performance goals. The DSPL framework is further generalized in Automatic Software Product Line (ASPL), which realizes self-adaptable products by combining automatic computing and self-adaptive software systems (Abbas 2011). However, this approach lacks an MDP-based learning approach that facilitates the optimization of transition decisions under uncertainty.

To specify properties of a system model, the authors in (Kwiatkowska, Parker, et al. 2011) introduces an incremental quantitative verification method based on an MDP formulation. The approach here takes advantage of the fact that the underlying MDP can be decomposed into its strongly connected components. To extend the capability of specifying possible transitions at design time to runtime, we note that reinforcement learning techniques can be useful for learning such possible transitions. In that context, tools from MDP can be used to our benefit to model reinforcement learning scenarios, especially in the domain of autonomic software (Amoui, Salehie, et al. 2008). For other formal methods including MDP

formulations in self-adaptive systems; we refer the reader to (Lemos, Giese, et al. 2013; Weyns, Iftikhar, et al. 2012).

The MDP framework provides policies for optimal control and has naturally been used in the context of controllable systems in communication networks. In this space, queuing theory provides useful abstractions that help us derive the building blocks of the underlying MDP. For applications of the framework to controlled queues, we refer the reader to (Crabill, Gross, et al. 1977; Ephremides and Verdu 1989; Stidham and Weber 1993). Authors in (Ephremides and Verdu 1989) use the MDP framework for network routing and multiple access control. An MDP can be reformulated as a linear program for optimization under certain conditions and this fact has been exploited in the work above in the context of the dynamic routing problem. The authors also consider the random access problem and formulate it using a controlled Markov chain model where the packet retransmission probability acts as the control variable.

The MDP framework has also been used in the context of congestion control, including in this thesis[1]. In (Hwang, Tan, et al. 2005), a Cooperative Multi-Agent Congestion Controller (CMCC) is proposed that can regulate the data traffic source flows by looking at the current environment conditions. Further, TCP Contention Control (TCP-CC), a TCP pacing protocol that aims to flatten TCP burstiness while maximizing the overall throughput, is also formalized using an MDP formulation (Xie and Boukerche 2015). In addition, the MDP framework has been used in the context of bandwidth adaptation problems for QoS provisioning in mobile communication networks (Fei, Wong, et al. 2006).

A section of literature on scheduling in wireless networks, for example, opportunistic scheduling (Asadi and Mancuso 2013; Ouyang, Murugesan, et al. 2015) has also used the MDP framework to analyze underlying systems. The hybrid Automatic-Repeat-Request (HARQ) protocol is analyzed using the MDP framework in the context of downlink wireless scheduling (Huang, Berry, et al. 2005) where the authors propose an optimal draining convex (DC) scheduling algorithm. The problem of multi-path data transfer in overlay networks has also been designed as a Markov decision problem in (Bui, Zhu, et al. 2010). The proposed solution, called Online Policy Iteration (OPI), solves the decision problem at runtime and achieves superior empirical performance to traditional JSQ and weighted round-robin policies.

In addition to proposing a general framework for modelling TBSs, we focus on specific classes of such systems and study the aspects of performance optimization therein. To that end, we formulated concrete research questions which we attempt to answer in this thesis. Recall from Sect. 1.1 that our first research question deals with the issue of performance optimization in parallel TBSs. In this context,

1 See Chap. 6 for details.

we initially focus on the abstraction of communication systems with multiple interfaces. In Chap. 3, we adopt a service-curve based approach to model the state evolution in such TBSs. To compare our approach to the existing results in similar systems, we first consider systems consisting of multiple components for providing service or extending service that vary over time. We first note that our service-curve based approach is fundamentally different from the well-known SCED and SCED+ algorithms which describe how a server can be shared between multiple sessions given a set of delay bounds and a measure of the burstiness of traffic arrival (Cruz 1998; Sariowan, Cruz, et al. 1999). In contrast, our approach proposes a policy to choose one of the alternative subsystems with an objective to reduce waiting times[2]. In this context, examples of optimization with respect to metrics other than delay include (i) optimization of the system throughput in wireless networks by minimizing buffer overflow probabilities for each session (Huang and Niu 2007), (ii) proposing a buffer-based uplink strategy in relay-assisted LTE networks to achieve a similar objective (Mehta, Khakurel, et al. 2012) and (iii) a queue-aware uplink scheduling that seeks to optimize resource utilization by asserting a given level of quality of service (QoS) simultaneously (Rizk and Fidler 2016).

While in our TBS, the service rate is not controllable and a decision is made based on the immediate performance implication of an action, a range of literature in this space focuses on systems with an adjustable service rate and measures their long-term performance. For example, the authors in (George and Harrison 2001) derive an optimal policy to control the service rate that leads to the optimal long-term average cost where average signifies cost per time. Similar approaches of dynamic control were also adopted in (Adusumilli and Hasenbein 2010; Wierman, Andrew, et al. 2008). While the former achieves its objective by adjusting the service at the beginning of each decision epoch, the latter adopts the approach of dynamic speed scaling to optimize a combination of the expected job response time and the expected energy cost.

In a setup extending a Markov modulated service like our TBS, the authors in (Mahabhashyam and Gautam 2005) analyze the average queue length, the waiting time, as well as the tail distribution of the workload. They further demonstrate the usefulness of their findings in the context of web servers with multi-class requests. In contrast our performance objective, we see that there is a body of work that considers the task of rate and power optimization. To minimize the average delay under power constraints, the authors in (Bettesh and Shamai 2006) consider a class of power and rate policies and optimize over them. A similar approach was taken in (Collins and Cruz 1999) to propose an optimal service policy to minimize the average power expended. The setup assumes that the

2 Although the metric can be generalized to similar measures of performance.

channels are of Gilbert-Elliot type and that power and rate are linearly related. Further, adopting the MDP framework, the authors in (Berry and Gallager 2002) propose a policy to minimize the long-term average transmission power and the average delay in fading channels.

To answer our first research question on transitions in parallel systems, we also consider TBSs consisting of heterogeneous groups of servers and a single dispatcher in Chap. 4. To model system evolution and subsequent performance of such TBSs, we adopt the popular setup of the supermarket model (Mitzenmacher 2001). This model provides an abstraction for many of today's datacenters where the number of servers is typically large. This setup renders policies that require information on the backlog at all (or many) servers infeasible. Thus, a randomized version of popular policies such as join the shortest queue (JSQ) is implemented in such TBSs. Under a randomized policy, the dispatcher samples a given number of server backlogs to schedule an incoming job. Despite the reduced overhead, such policies were seen to yield comparable performance to their classical equivalent (M. Mitzenmacher 1996). A majority of the randomized policies are JSQ-based and we generalize these policies by introducing the notion of cost. Accordingly, we call our policy cost-based scheduling (CBS), which has the advantage of encoding the suitability of a group of servers with regards to a particular job. In the following, we review the existing literature in the context of the framework we adopt to model the current TBS, i.e., the supermarket model.

CBS policies can be viewed as a generalization of JSQ-type randomized policies and, consequently, the majority of randomized policies, such as the schemes 1 and 2 presented in (Mukhopadhyay, Karthik, et al. 2016), the power of two policy (Mitzenmacher 2001; Vvedenskaya, Dobrushin, et al. 1996) and the subsequent generalization, the SQ(d) policy (Godtschalk and Ciucu 2016; Mukherjee, Borst, et al. 2016), can be obtained as special cases of CBS policies. A thorough survey of SQ(d)-type policies is provided in (Boor, Borst, et al. 2018). The first study on the number of servers sampled d in SQ(d) policies, in some sense, was (Mitzenmacher 2001) where it was shown that the expected time spent in a supermarket model with N servers improves exponentially when d is increased from 1 to 2 in the asymptotic regime ($N \to \infty$). However, when d is increased from 2 to 3, the expected response time improved merely by a constant factor. Incidentally, the asymptotic estimate becomes less accurate in the finite regime and certain upper and lower bounds on the average delay can be found in (Godtschalk and Ciucu 2016). Aside from results on delays in the supermarket model, a bound on the queue lengths is provided in (Brightwell, Fairthorne, et al. 2018) under appropriate scaling.

To measure performance in the TBS at hand, we first derive a scaling limit of the tail distribution of server backlogs which enables tractable performance evaluation. A related result involving a scaling limit was derived in (J. G. Dai and

Meyn 1995), where the authors considered the issues of stability and convergence of certain moments of the queue length in a multi-class queuing system. In contrast to our TBS, where servers are assumed to have finite buffers as observed in reality, this work considers servers with infinite buffers. A later paper by the authors (J. Dai and W. Dai 1999) considers a finite-buffer open queuing network with deterministic and feedforward routing where a Semi-Martingale Reflecting Brownian Motion (SRMB) limit was derived for an appropriately scaled queue length process. Additionally, an extension of the results concerning heavy traffic limit theorems was discussed. A recent result (Mukherjee, Borst, et al. 2016) focuses on the SQ(d) policy in large-scale systems with identical servers. The authors derive fluid and diffusion limits for scaled buffer occupancies as the number of servers becomes arbitrarily large and show that these limits under the SQ(d) scheduling policy correspond to that of JSQ under certain conditions. Further, the paper (Mukhopadhyay and Mazumdar 2016) considers a TBS similar to ours, although with infinite-buffer queues, and investigates the aspect of stability under randomized JSQ scheduling. Using a scaling limit, the authors demonstrate that the uniform sampling of servers may lead to a reduced stability region when the clusters are heterogeneous. JSQ scheduling was further analyzed in (Banerjee and Mukherjee 2019, 2020) where a parallel single-server queueing system was considered. The papers focus on the stationary distribution of the occupancy measure of the system and estimates are derived for the tail-asymptotics and the bulk behaviour.

In our analysis of the TBS with a single dispatcher and groups of servers, we assume exponentially distributed interarrival and service times for tractability. We further derive a scaling limit to analyze the performance of the TBS. We use the popular method of obtaining scaling limits for density-dependent Markov jump processes, which relies on Kurtz's approximation theorems (Ethier and Kurtz 1986). However, for systems with non-exponentially distributed arrival and service times, this approach no longer works. In such non-Markovian settings, useful scaling limits can still be derived under certain restrictions; see (Aghajani, X. Li, et al. 2017; Aghajani and Ramanan 2017; Decreusefond and Moyal 2008; Gromoll, Puha, et al. 2002) for details.

A common assumption in the queueing literature is that the considered queues are infinite, which is motivated by the tractability of the resulting framework. However, many queueing systems in the real-world have limited buffers and, in some cases, are not well-modelled by the infinite-buffer formulation (Ciucu, Poloczek, et al. 2019b). Further, infinite-buffer models cannot directly measure the impact of the buffer size on any given QoS metric. This motivates us to incorporate the finiteness of buffers into our abstraction of the TBS. To get an overview of the developments in this domain, we refer readers to the surveys (Balsamo, Personé, et al. 2003; Perros 1989). A host of results on finite-buffer

systems aims to derive performance bounds by appropriately approximating infinite-buffer systems. For example, in (Kim and Shroff 2001), the total loss probability in a finite-buffer server with a constant service rate was approximated by the tail probability of the queue length distribution assuming the buffer is infinite. Some of the distinct approaches in this sphere include: (i) the SRBM limit obtained in (J. Dai and W. Dai 1999) under the finite-buffer assumption, which can be used to find stationary queue length distributions, (ii) the finite element method for computing the stationary distribution in finite-buffer systems (Shen, Chen, et al. 2002) and (iii) decomposition methods for approximate performance evaluation (Gershwin 1987; Sadre, Haverkort, et al. 1999).

Our second research question focuses on the aspect of performance optimization in closed batch-processing TBSs. We model such TBSs in Chap. 5 using a closed queueing network comprising multiple stations. Batching in such TBSs is motivated by reduced operational overhead and we quantify the benefit using a speedup function that takes mean service times for different batch sizes as inputs. Since the batching phenomenon is a defining characteristic of such systems and, naturally, their performance, we present an overview of batching in open and closed queueing systems.

In open queueing systems, service batching was considered as early as 1954, and the work (Bailey 1954) derived the mean steady-state queue length and waiting time under the assumption of exponentially distributed interarrival and Chi-squared service time. Further, a queueing system with exponentially distributed interarrival and general batch service time is considered in (Deb and Serfozo 1973). Here, the service time distribution is independent of the batch size and the batch size, as well as the execution time of the batch, can be dynamically controlled without breaching certain real-world constraints. The authors show that if a batch can be submitted for service only when the server is free, or during an arrival or departure event, it is optimal to serve all jobs in a batch only if the queue filling goes beyond a certain cut-off. Service batching has been considered to optimize other metrics of interest as well. For example, in the context of running a shuttle service, batching was employed to minimize the expected total long-term discounted cost (Deb 1978). The authors in (Berg, Duyn Schouten, et al. 1998) also consider batching to optimize a similar metric. In a discrete time system serving jobs with hard delay guarantees, the authors propose a policy to minimize the expected long term cost per unit time for a given form of serving cost. We further note that if we assume the batching step in our TBS to be instantaneous, the particular dynamics in the service station becomes equivalent to a system with batch arrival and batch service. However, the interarrival times under this assumption no longer follow an exponential distribution and the model loses tractability. A similar model, although under the tractable assumption of exponential batch arrival and service, is analyzed

in (Germs and Foreest 2013). Unlike our TBS, the system in this case discards jobs beyond a certain queue length threshold. We conclude the comparison of batching in open system by noting that they are inherently different than their closed counterparts. A common assumption in open systems is independence of arrival and service processes which, in most cases, cannot account for the dynamics of closed system where jobs are routed within different stations.

Irrespective of batching, a central idea for analyzing closed queueing systems is to derive a product form of the steady-state distribution. For foundational results in this domain, readers are referred to (Baskett, Chandy, et al. 1975; Chandy, Howard Jr, et al. 1977; Chandy and A. J. Martin 1983; Gordon and Newell 1967). To the best of our knowledge, (Henderson, Pearce, et al. 1990) was the first work to achieve a significant result on the product form of the stationary distribution for closed batching systems. The authors in this paper derive the conditions for the existence of the product form in a system that operates in discrete time. Further, the jobs follow state-independent routing and the system allows occurrence of multiple events in a single time slot. The authors extend their results to accommodate for batch arrivals in (Henderson and Taylor 1990), although in a continuous-time setting. This work comes closest to our analysis of the closed batch-processing TBS. However, it does not readily apply to our case. This is because the conditional routing probabilities of the jobs or the batches in our case are state-dependent as the TBS follows FCFS service discipline. Further, a limiting feature of the above work is that it does not specify a method to derive the normalizing constant of the product form distribution, which makes it ineffective for immediate performance analyses.

Next, we compare our method to some of the well-known approaches to batching in practice. A typical example of a real-world batching system is a database employing query batching. One of the earliest examples of query batching in databases was (DeWitt, Katz, et al. 1984) where authors focused on I/O reduction. The batch size in this work was fixed to page size, whereas in our TBS, the batch size can be recalculated following a system reconfiguration. Further, unlike our system, which uses a fixed yet configurable batch size to optimize system throughput, *SharedDB* (Giannikis, Alonso, et al. 2012) aggregates all jobs into a single batch while a previously-formed batch is being executed. Thus, in contrast to our TBS, the batch size for *SharedDB* is variable and, additionally, it is agnostic of job types. *BatchDB*, which is similar to SharedDB, also shares such differences with our approach. The work that resembles our approach the most is *OLTPShare* (Rehrmann, Binnig, et al. 2018), where incoming jobs in a fixed time interval are accumulated to form a batch. In contrast, we use count-based batching (i.e., a fixed batch size that can be changed upon system reconfiguration), which has the advantage of utilizing cached batch queries.

Having considered transitions in specific classes of TBSs, we focus on the aspect of overall networked system quality in our third research question. This is particularly important given the dependence of performance of the connected TBSs on the basic functionalities of the network. In that context, we consider the problem of Active Queue Management (AQM) in network switches in Chap. 6. AQM tries to contain bufferbloat (Gettys 2011), the phenomenon of excess buffering at a switch. AQM has been introduced in the early 90s and the earliest example of AQM is Random Early Detection, or RED (Floyd and Jacobson 1993). To prescribe a policy as to when a packet should be dropped, RED takes the current queue length as an input. The policy depends critically on two threshold parameters for the queue length. When the current queue-filling is below the lower threshold, a packet is always admitted whereas a queue length beyond the higher threshold leads to packet drop. For an intermediate queue-filling, RED calculates a drop probability which dictates the action of packet drop. Although RED is shown to be fairer to bursty traffic than the classic drop tail, the main challenge lies in identifying the threshold parameters and there has been no consensus to the corresponding parameter engineering process. To get around this problem, a range of variants and extensions of RED has been proposed, which addressed this issue with fairly limited success. For a thorough discussion on these algorithms, we refer to (Adams 2013) and the references therein. The performance of a switch employing RED has also been analyzed from the perspective of control theory (Hollot, Misra, et al. 2001b) and subsequently compared it with related control-theoretic approaches (Hollot, Misra, et al. 2001a). Further, jump process driven SDEs were used to model the interaction between multiple TCP flows and an AQM-enabled switch implementing RED in (Misra, Gong, et al. 2000).

The AQM algorithm that has successfully circumvented the problem of parameter engineering is CoDel (Nichols and Jacobson 2012). Although CoDel is driven by two input parameters, one that signifies the target delay and the other specifying how often a packet should be dropped, their default value is hardly changed in practice. Thus, CoDel is considered to be parameterless. Starting with the default value of the window parameter that signifies the frequency of packet drop, CoDel successively adapts its value until it meets the target delay. Since an AQM-enable switch encounters diverse traffic conditions, the choice of the default values of the input parameters requires careful considerations and there has not been much research in this space.

While CoDel drops a packet after it is already enqueued, the algorithm PIE (Pan, Natarajan, et al. 2013a) propose to overcome this additional processing overhead by dropping the packet at the input port. PIE calculates the drop probability of an incoming packet by looking at the current queue-filling and the departure rate from the queue. This improves the processing overhead to some extent compared to CoDel, which requires calculation of delay per packet. Further, the

consideration of the current queue length to calculate the drop probability implies that congestion is directly controlled. Although claimed to be knob-free like CoDel, PIE actually requires default values for target delay, drop frequency and parameters for drop probability adaptation (Khademi, Ros, et al. 2014). Further, the drop probability parameters are adapted according to a rule that is based on judgements. PIE and CoDel have been compared extensively against each other and to some variants of RED (Järvinen and Kojo 2014; Khademi, Ros, et al. 2014). The former paper considers both performance and and its scalability under multiplexing. It concludes that CoDel visibly leads to better performance in terms of delays while the performance of PIE scales well for multiplexed flows. Similarly, the authors (Khademi, Ros, et al. 2014) recommend CoDel over PIE after their extensive evaluation over the range of respective default parameters as the empirical delay distribution under PIE was seen to have a longer tail.

In light of the above, we propose a principled approach to address the AQM problem which only requires the target delay as an input. We take a direct approach that uses the current queue-filling and the arrival rates to suggest a drop/admit action. Our policy is built using the MDP framework where we optimize for a precise performance criterion combining delay and throughput. In Chap. 6, we review our approach against CoDel and the classic drop tail queues where we see that the proposed policy yields equivalent throughput to CoDel while minimizing delays considerably.

3 DELAY OPTIMIZATION VIA EPOCH-BASED TRANSITIONS

3.1 INTRODUCTION

In this chapter, we focus on our first research question which considers the problem of performance optimization in open transition-based systems (TBSs) employing parallelism. Specifically, we consider systems where transition is triggered by time. Such a TBS was shown at the bottom of Fig. 1.1. Recall that the TBS was equipped with multiple communication technologies that could be switched periodically at runtime in order to minimize the delay. Such a system provides an abstraction for modern mobile devices that have access to multiple wireless technologies such as LTE, WiFi, Bluetooth and mmWave. Each of these technologies is influenced by widely varying and diverse channel conditions and has significant differences in the PHY and MAC layer implementations. Our formulation of the TBS seeks to benefit from the diversity of these technologies offering statistically independent service and their appropriateness given the current system state.

As mentioned, the TBS switches between the available communication technologies at the beginning of each period. A period has a fixed time length and thus a transition is triggered by time. During each transition, the system analyzes the assured level of quality of service/performance until the next transition and decides on the most suitable communication interface for that period. Consequently, we refer to the period or the transition frequency of the TBS as decision epoch. We want to emphasize the fact that the TBS uses a communication interface exclusively for a decision epoch, i.e., there is no multipath communication.

The objective of the TBS is to devise a policy that minimizes the delay at the end of the decision epoch. We also investigate the long-term behaviour of the policy empirically in Sect. 3.4. Although we choose delay as the performance metric for our analysis, it is possible to customize the policy to account for a different tractable metric such as throughput.

A schematic description of the TBS is shown in Fig. 3.1. Note that on a flow level, the i-th subsystem, i.e., the communication technology, provides a time-varying service S_i to the flow arrivals A that are to be transmitted. For the given metric for the communication quality, i.e., delay, the TBS alternates between available subsystems to optimize the metric of interest. This implies that each subsystem receives a thinned stream from the common arrivals to the system.

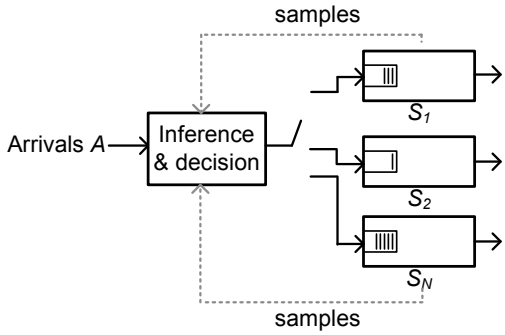


Figure 3.1: Transition-based system: subsystems extend different service $S_1, \ldots, S_N$ and the system picks a subsystem at the beginning of every epoch to transmit the arrivals A. The choice of the subsystem is based on the current system state and is aimed at achieving optimal performance at the end of the decision epoch.

In the following, we provide a formal framework for the considered TBS using a service curve description (Ciucu, Burchard, et al. 2006) for different subsystems. At the same time, our formulation factors in the existing backlogs at different subsystems. More specifically, the TBS determines the most appropriate subsystem at the beginning of every epoch of length Δ based on delay bounds for different subsystems $i \in \{1, \ldots, N\} = [N]$. Thus, according to the terminology introduced in Chap. 1, the system state at t-th decision epoch S_t constitutes of the backlogs at each interface at the beginning of the epoch and the context ξ_t is given by the service characteristics of each interface $i \in [N]$. Further, the action Λ_t denotes the index of the subsystem chosen for that decision epoch, which implies that the action set $\mathcal{A} = [N]$. Note that the system can also make a self-transition, which implies picking the same subsystem over two consecutive epochs. At state $\mathsf{X}_t = (\mathsf{S}_t, \xi_t)$, the cost of the transition[1] $R(\mathsf{X}_t, \Lambda_t)$ is defined as the maximum possible delay at the end of the current epoch. For the sake of simplicity, we do not adopt the MDP framework here to find the optimal policy for minimizing delay. Rather, we take a heuristic approach where the immediate cost $R(\mathsf{S}_t, \Lambda_t)$ is minimized. We formally introduce the quantities mentioned here in Sect. 3.3.1. Note that for unknown service characteristics of the subsystems, the TBS observes the service pattern and makes inference based on the available data.

To summarize, our contributions here are twofold: first, to be able to devise a policy for a wide class of service distributions, we provide an envelope for general discrete time Markov modulated service processes. Second, we formally introduce an adaptive assignment policy, which is seen to yield superior worst-case performance numerically for a system with heterogeneous multi-interfaces.

1 Recall that we use $R(\mathsf{X}_t, \Lambda_t)$ to denote cost *or* reward depending upon the context.

Here, we choose the maximum waiting time across the subsystems as the metric for worst-case performance. However, we note that the edge of the adaptive assignment policy over considered alternatives, measured in terms of maximum and average waiting times, tends to shrink with diminishing system utilization.

We structure the rest of the chapter as follows: the queueing model that forms the basis for the TBS is elaborated in Sect. 3.2. Subsequently, service models for wireless systems are reviewed in Sect. 3.3. The performance and related aspects of the adaptive policy are compared numerically to relevant alternatives in Sect. 3.4.

3.2 QUEUE AWARE SCHEDULING

First, we define the queue aware scheduling system where time is assumed to be discrete. We denote the arrivals to the system between time points $s+1$ and u by $A(s,u)$ whereas the departures of the system in the same period are denoted by $D(s,u)$. The time-varying service offered by the system in the period $\{s+1,\ldots,u\}$ is modelled using the random process $S(s,u)$. Further, we define $A(u) = A(0,u)$ and $D(u) = D(0,u)$, which let us write the backlog at time u as $B(u) = A(u) - D(u)$. Using the notion of dynamic server (Chang 2000), the departures can be written in terms of the arrivals and the service as

$$D(u) \geq \min_{s\in[0,u]} \{A(s) + S(s,u)\} .$$

This leads to a bound on the backlog of the form

$$B(u) \leq \max_{s\in[0,u]} \{A(s,u) - S(s,u)\} .$$

The stochastic analogue of the above bound introduces the concept of violation probability. This implies that the bound can be breached with probability ϵ, typically small. For the discrete time model, the computation of the violation probability involves making a sample path argument that is typically based on the union bound (Ciucu, Burchard, et al. 2006; Cruz 1996; Fidler and Rizk 2015).

As mentioned, the available communication technologies at the TBS have inherently different implementation and hence offer different service guarantees. The devised policy switches between alternate technologies and the switching rule relies on the service curve description thereof. To be specific, at the beginning of each epoch t, i.e., at time $t\Delta$, with $t \in \mathbb{Z}_+$ and epoch length Δ, the decision making unit of the system chooses the subsystem Λ_t among available subsystems with service processes $\{S_1, S_2, ..., S_N\}$, such that

$$\Lambda_t = \arg\min_{i\in[N]} w_{i,t} , \quad \text{with}$$
$$w_{i,t} = \inf\{w : \mathsf{P}\left(W_{i,t} > w_{i,t}\right) \leq \epsilon\} .$$

Here, $W_{i,t}$ denotes the waiting time at the end of the t-th epoch if the i-th subsystem is chosen and ϵ stands for a predefined service quality expressed through a violation probability. We also look at the assignment problem when a characterization of the service processes $\{S_1, S_2, ..., S_N\}$ is not directly available. In this case, the decision module of the system draws samples from the observed service to infer the parameters that describe the service curves as shown in Fig. 3.1.

3.3 PROPERTIES OF THE TRANSITION-BASED SYSTEM

In our formulation, we assume that the arrivals and service processes admit envelopes similar to (Ciucu, Burchard, et al. 2006; Cruz 1996; Fidler and Rizk 2015; C. Li, Burchard, et al. 2007; Yin, Jiang, et al. 2002), i.e., we consider arrivals yielding an $(b_A, \rho_A, \epsilon(b_A))$ envelope of the form

$$\mathsf{P}\left(\exists s \in [0,u] : A(s,u) > \rho_A(u-s) + b_A\right) \leq \epsilon(b_A)\ . \tag{3.3.1}$$

Further, we consider service processes that adhere to $(-b_i, \rho_i, \epsilon(b_i))$ envelopes in the sense of (Fidler and Rizk 2015), i.e.,

$$\mathsf{P}\left(\exists s \in [0,u] : S(s,u) < \rho_i(u-s) - b_i\right) \leq \epsilon(b_i)\ , \tag{3.3.2}$$

for all subsystems $i \in [N]$. Observe that the service process envelopes satisfy the notion of service curves. A more compact service envelope is given by $\max\{\rho_i(u-s) - b_i, 0\}$ as the service process is always non-negative.

3.3.1 *Transition Scheme*

Assuming that the arrival and service processes admit envelopes $(b_A, \rho_A, \epsilon/2)$ and $(-b_i, \rho_i, \epsilon/2)$, respectively, satisfying $\rho_i > \rho_A, \forall i$, the delay $W(s)$ at time s satisfies:

$$\mathsf{P}\left(W(s) > w\right) \leq \epsilon\ ,$$

where $w = (b_A + b_i)/\rho_i$ (Fidler and Rizk 2015).

Using this bound, we define the cost function R and derive an adaptive policy that picks the subsystems with minimum cost every decision epoch. For the TBS, the state at t-th decision epoch is given by $\mathsf{X}_t = (\mathsf{S}_t, \xi_t)$ where the system state is given by the vector of backlogs, i.e., $\mathsf{S}_t = (B_{1,t}, \ldots, B_{n,t})$ and the context is given by the collection of arrival and service envelope parameters: $\xi_t = (b_A, \rho_A, -b_1, \ldots, -b_n, \rho_1, \ldots, \rho_n, \epsilon/2)$. Note that the context does not change if the service characteristics of the communication interfaces are known a priori. Using

the context and system description, we define the cost function as the waiting time bound of the chosen interface/subsystem, which can be expressed as:

$$R(\mathsf{X}_t, \Lambda_t) = w_{\Lambda_t,t} = \frac{(b_A + B_{\Lambda_t,t} + b_{\Lambda_t})}{\rho_{\Lambda_t}} .$$

This is based on the fact that we can treat the existing backlog of a subsystem as additional arrivals at the first time point in the epoch to reconstruct the envelope. Thus, the index i_t^* of the system chosen at the beginning of the epoch can be expressed as

$$i_t^* = \arg\min_{\Lambda_t \in \mathcal{A}} R(\mathsf{X}_t, \Lambda_t) = \arg\min_{i \in [N]} \frac{b_A + B_{i,t} + b_i}{\rho_i} . \tag{3.3.3}$$

We drop the epoch index t from i_t^* when the dependence is clear. Note that our system state description implicitly assumes that each backlog $B_{i,t}$ is observable at the beginning of epoch t. To cite an example of an envelope, for arrival and service processes following independent Poisson processes with parameters λ and μ_i, respectively, it holds that $\big(-\log\epsilon, \lambda(e^{\theta}-1)/\theta, \epsilon\big)$ and $\big(\log\epsilon, -\mu_i(e^{-\theta}-1)/\theta, \epsilon\big)$ are valid envelopes. Here, $\theta > 0$ is a free parameter that can be optimized.

In the following subsections, we derive envelopes when the service increments are Markov modulated. Recall that the envelope for each subsystem is derived separately. Thus, the derivations are in the context of a given subsystem, which can be replicated for other subsystems as well. In that spirit, we drop the subsystem index i in these subsections for the sake of brevity.

3.3.2 *Markov Modulated Service*

In this subsection, we focus on the specific case of wireless fading channels where we borrow the finite state Markov chain framework for modelling the service increments from (Sadeghi, Kennedy, et al. 2008). We assume that the service increment X_j at time slot j is modulated by a Markov chain $\{Z_j\}_j$ where $Z_j \in \{0,1\}$. Denoting the conditional distributions of the increment as $(X_j|Z_j = 0) \sim Y_0$ and $(X_j|Z_j = 1) \sim Y_1$ and the corresponding moment generating functions $\mathsf{E}\left[e^{\theta Y_k}\right] = M_k(\theta)$, $k \in \{0,1\}$; we see that

$$\begin{aligned}\mathsf{E}\left[e^{\theta S(s,s+u)}|\mathbf{Z}\right] &= \prod_{j=s+1}^{s+u} \left(M_0(\theta)\right)^{1-Z_j} \left(M_1(\theta)\right)^{Z_j} \\ &= \big(M_0(\theta)\big)^{u-\sum_{j=s+1}^{s+u} Z_j} \big(M_1(\theta)\big)^{\sum_{j=s+1}^{s+u} Z_j} ,\end{aligned}$$

where $\mathbf{Z} = (Z_{s+1}, \ldots, Z_{s+u})$. We would like to point out the fact that the emission distributions, i.e., the conditional distributions of X_j given Z_j are independent of

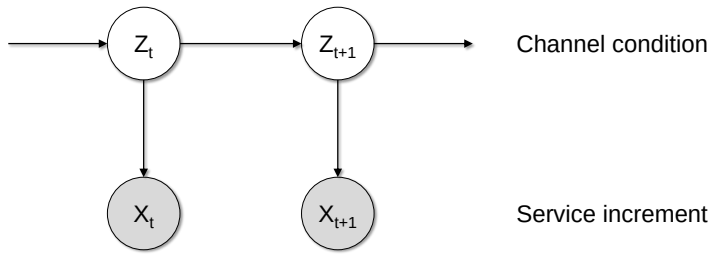


Figure 3.2: A schematic representation of Markov-modulated service: the underlying channel condition Z_t at epoch t for the considered subsystem has a memory as represented by an arrow between them. The channel condition Z_t influences the actual service increment in that epoch, given by X_t. An example is given in Sect. 3.4.2 where the channel of the considered subsytem is in either *On* or *Off* state, i.e., $Z_t \in \{0,1\}$ and $X_t|Z_t$ is assumed to follow an exponential distribution.

each other and independent of time, which explains the absence of the time index j in the variables Y_0 and Y_1. Now,

$$\begin{aligned}
\mathsf{E}\left[e^{-\theta S(s,s+u)}\right] &= \mathsf{E}\left[\mathsf{E}\left[e^{-\theta S(s,s+u)}|\mathbf{Z}\right]\right] \\
&= \mathsf{E}\left[\left(M_0(-\theta)\right)^{u-\sum_{j=s+1}^{s+u} Z_j}\left(M_1(-\theta)\right)^{\sum_{j=s+1}^{s+u} Z_j}\right] \\
&= \sum_{\boldsymbol{z}\in\{0,1\}^u}\left(M_0(-\theta)\right)^{u-\boldsymbol{z}'\mathbf{1}}\left(M_1(-\theta)\right)^{\boldsymbol{z}'\mathbf{1}} p_u(\boldsymbol{z}) \\
&\leq \left(\sum_{\boldsymbol{z}\in\{0,1\}^u}\left(M_0(-\theta)\right)^{lu-l\boldsymbol{z}'\mathbf{1}}\left(M_1(-\theta)\right)^{l\boldsymbol{z}'\mathbf{1}}\right)^{1/l}\left(\sum_{\boldsymbol{z}\in\{0,1\}^u} p_u^m(\boldsymbol{z})\right)^{1/m} \\
&\leq \left(M_0^l(-\theta)+M_1^l(-\theta)\right)^{u/l}\left(\tilde{p}_u\right)^{1/l}.
\end{aligned} \tag{3.3.4}$$

Here $\boldsymbol{z}'\mathbf{1} = \sum_j z_j$, $p_u(\boldsymbol{z}) = \mathsf{P}\left((Z_{s+1},\ldots,Z_{s+u}) = \boldsymbol{z}\right)$, and

$$\tilde{p}_u = \max_{\boldsymbol{z}\in\{0,1\}^u} \mathsf{P}\left((Z_{s+1},\ldots,Z_{s+u}) = \boldsymbol{z}\right).$$

Observe that the first inequality in (3.3.4) is a Hölder's inequality where m and l satisfy $1/m + 1/l = 1$ and $m > 1, l > 1$. The second inequality is based on the fact that $p_u^m(x) \leq p_u(x)(\tilde{p}_u)^{m-1}$ and substitutes m in terms of l. Further, $\theta > 0$ and $l > 1$ are two parameters that can be optimized.

The bound (3.3.4) can be further generalized for the case where the modulating variable Z_j has K states. Denoting corresponding moment generating functions $\mathsf{E}\left[e^{\theta X_j}|Z_j = k\right] = M_k(\theta)$, $k \in [K]$; we see that

$$\mathsf{E}\left[e^{-\theta S(s,s+u)}\right] \leq \left(\sum_{k=1}^{K} M_k^l(-\theta)\right)^{u/l} \left(\tilde{p}_u\right)^{1/l} , \tag{3.3.5}$$

where θ, l and $\tilde{p}_u$ are same as in (3.3.4).

Remark. *The bound in (3.3.5) lends itself nicely to calculate envelopes of the form $(-b_i, \rho_i)$ as in (3.3.2) for MIMO wireless systems under spatial multiplexing. Here, the K states represent the available degrees of freedom due to multiple antennas.*

3.3.2.1 *Markov Modulated On-Off Service*

The simplest and most ubiquitous special case of Markov modulated service from Sect. 3.3.2 is Markov modulated On-Off service where the modulating factor can only be in *On* or *Off* state. Here, $Z_j = 0$ and $Z_j = 1$ represent the *Off* and the *On* states, respectively. Evidently, there is no service increment in the *Off* state, i.e., $Y_0 \equiv 0$, whereas constant rate service increments provided by the channel in the *On* state is denoted as $Y_1 \equiv r$. Plugging in Y_0, Y_1, and $l = 2$ in (3.3.4) gives

$$\mathsf{E}\left[e^{-\theta S(s,s+u)}\right] \leq \left(1 + e^{-2\theta r}\right)^{u/2} \sqrt{\tilde{p}_u} .$$

Further, we have $\tilde{p}_u \leq \max(\mathbf{P})^{u-1}$, where $\mathbf{P}$ denotes the one-step transition matrix of $\mathbf{Z}$. Denoting $p_* = \max(\mathbf{P})$, we have

$$\mathsf{P}\left(S(s,s+u) < E_s(u)\right) \leq e^{\theta E_s(u)} (1 + e^{-2\theta r})^{u/2} p_*^{(u-1)/2} . \tag{3.3.6}$$

Further, defining ϵ_s as the violation probability on the right hand side of (3.3.6), the service envelope $E_s(u)$ can be written as

$$E_s(u) = \frac{1}{\theta}\left(\log \epsilon_s + \frac{\log p_*}{2} - u\left[\frac{\log(1 + e^{-2\theta r})}{2} + \frac{\log p_*}{2}\right]\right) , \tag{3.3.7}$$

where $\theta > 0$ is a free parameter to be optimized. If not known a priori, the parameter p_* can be inferred from observed service increments, which we describe next.

Let us denote the underlying Markov chain of the On-Off process as $\mathbf{Z}$. Hence at time j, $Z_j = X_j/r$, i.e., the state of the underlying Markov chain is directly deducible from the observed service increments. Let $\mathbf{P}$ be the 2×2 transition matrix of the chain:

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} .$$

Recall that we restrict our attention to a fixed subsystem i for deriving the corresponding service envelope. Since our algorithm uses the same subsystem for an entire epoch, for this estimation, we can just pick the epochs where the i-th subsystem was used for transmission. Let us denote the corresponding service increments as $\{X_\beta^\alpha\}_{\alpha,\beta}$, where X_β^α denotes the service increment at β-th time slot of an epoch when the i-th subsystem was chosen for the α-th time. If there are n time slots in this epoch, we have

$$\hat{p}_{00} = \frac{\sum_\alpha \sum_{\beta=1}^{n-1} \mathbb{1}_{X_\beta^\alpha>0} \mathbb{1}_{X_{\beta+1}^\alpha>0}}{\sum_\alpha \sum_{\beta=1}^{n-1} \mathbb{1}_{X_\beta^\alpha>0}}.$$

Estimation of other elements of $\mathbf{P}$ proceeds in a similar fashion. The maximum of these estimates then can be plugged in (3.3.7). Note that for the general case where the modulating chain has K levels, one can use estimate the corresponding transition matrix using standard estimation algorithm for Hidden Markov Models (HMM); see Chap. 13 of (Bishop 2006) for details.

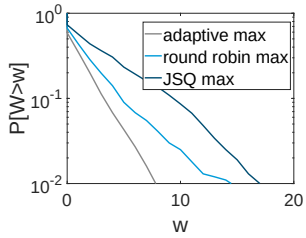
Remark. *The On-Off service model in Sect. 3.3.2.1 is known as the Gilbert-Elliott channel model (Gilbert 1960) for wireless fading channels. Here, a channel can be either in a good (*On*) or in a bad state (*Off*). Evidently, transmission is possible only in the good state. The* On *rate r and the channel memory, modelled using the transition matrix $\mathbf{P}$, are the parameters that characterize the model.*

3.4 NUMERICAL EVALUATION

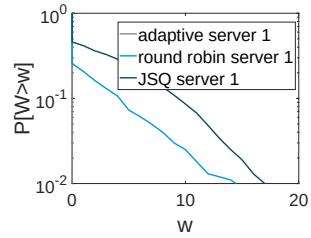
In this section, we compare the performance of our adaptive algorithm that uses the selection rule in (3.3.3) to round-robin (RR) and join the shortest queue (JSQ) subsystem assignment. Note that for JSQ, the expression in (3.3.3) reduces to choosing the subsystem $i^* = \arg\min_{i\in[N]} B_i$ at the beginning of every epoch. We focus on two distinct cases: (i) subsystems extending memoryless Poisson service and (ii) subsystems having channels with memory, which is amenable to the MMOO model. For the sake of simplicity, we assume that the system arrivals follow a Poisson distribution. Unless stated otherwise, the number of simulation runs for each experiment is set to 10^3, each ranging over 10^4 time slots, and the decision epoch length is fixed at 10 slots.

3.4.1 *Poisson Service*

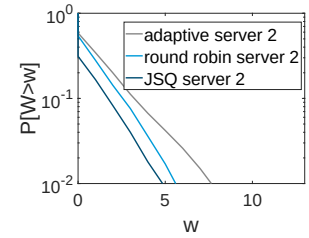
This subsection deals with the case when both the arrival and subsystem service increment processes follow a Poisson distribution. More specifically, the arrival and the service increments are iid Poisson with parameters λ and $\mu = [\mu_1, \ldots, \mu_N]$, respectively. That is, μ_i denotes the average service increment rate of i-th subsystem.



(a) Empirical CCDF of the maximum of the waiting times at two subsystems.

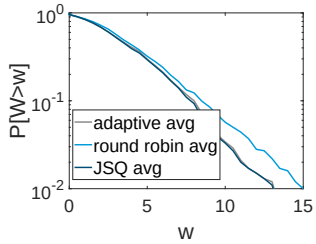


(b) Empirical CCDF of the waiting times at the slower subsystem.

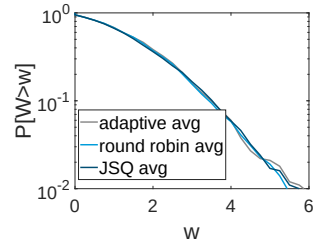


(c) Empirical CCDF of the waiting times at the faster subsystem.

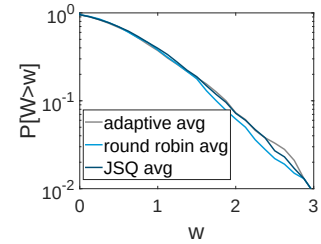
Figure 3.3: Two subsystems with mean service rates $\mu = [1, 2]$. At the slower subsystem, the empirical CCDF of the adaptive policy coincides with the y-axis. Simulation parameters: arrival rate $\lambda = 1.5$, violation probability $\epsilon = 10^{-4}$.



(a) $N = 2$ subsystems.



(b) $N = 4$ subsystems.

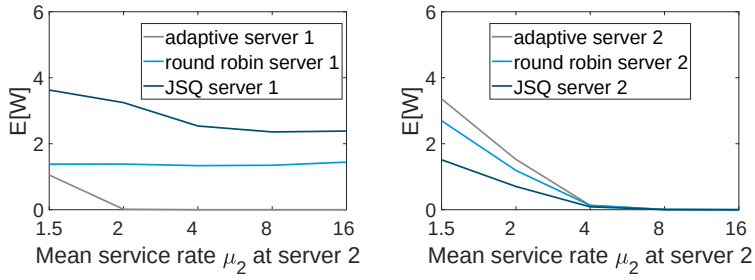


(c) $N = 8$ subsystems.

Figure 3.4: Empirical CCDF of the average waiting time at N subsystems: strong influence of the number of subsystems N on the average waiting time in the system. All three strategies show comparable performance. Arrival rate $\lambda = 1.5$, mean service rates $\mu_i = 1$ for every subsystem i, violation probability $\epsilon = 10^{-4}$.

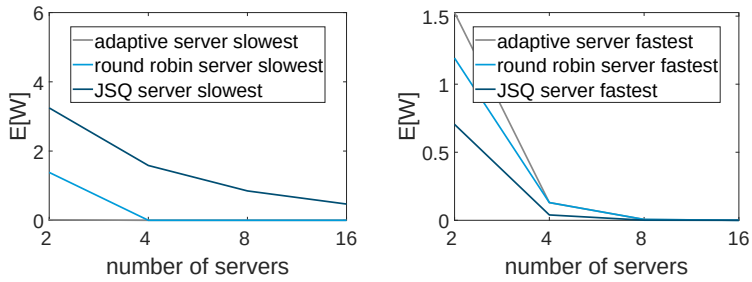
We explore the influence of the following factors on system performance: (i) heterogeneity of subsystems, (ii) total number of subsystems N and (iii) the epoch length Δ. We plot CCDFs of the waiting times at different subsystems, as well as, the CCDF of the maximum waiting time across different subsystems. Through these plots, we intend to highlight the effect of the assignment policy on the worst-case waiting time in the system.

To begin with, Fig. 3.3 shows the CCDF of the maximum waiting time and at the same time the subsystem-specific waiting times. The underlying system has $N = 2$ subsystems with heterogeneous service rates $\mu = [1, 2]$ and the arrival rate is $\lambda = 1.5$. This scenario of two heterogeneous subsystems can be thought of as two technologies such as WiFi and LTE, having different average bandwidths with the convenient assumptions of iid service increments. Our figures highlight the fact that in the heterogeneous case, the adaptive policy enhances the *worst-*



(a) Slower subsystem **(b)** Faster subsystem

Figure 3.5: The expected waiting time at subsystems 1 and 2 for a varying mean service rate of the second subsystem. Here, $\mu_1 = 1$, $\lambda = 1.5$, $\epsilon = 10^{-4}$ and the number of simulation runs equals 10^4 .



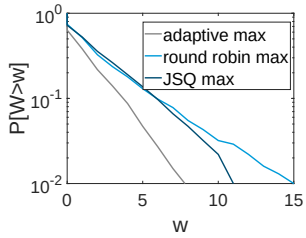
(a) Slowest subsystem **(b)** Fastest subsystem

Figure 3.6: The expected waiting time at the slowest and the fastest subsystem for increasing number of subsystems N. Here, subsystem i possesses a mean service rate $\mu_i = i$. Further, we fix $\mu_1 = 1$, $\lambda = 1.5$, $\epsilon = 10^{-4}$ and the number of simulation runs to 10^4 .

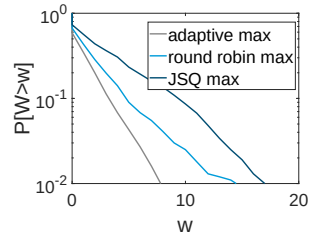
case performance, measured by the maximum of the waiting times of the two subsystems. Looking at Fig. 3.3c we see that this gain comes at the cost of longer waiting times at the faster system.

In Fig. 3.4, we aim to quantify the performance gain due to the introduction of a new subsystem. We plot the CCDFs of average waiting times for an increasing number of *homogeneous* subsystems N, each extending a mean service rate $\mu_i = 1$. Interestingly, we see a comparable performance of the average system waiting times for all three candidate strategies, i.e., RR, JSQ, and the adaptive assignment. This leads us to the conclusion that differences between the assignment strategies shrink with decreasing utilization, and in homogeneous settings, the adaptive policy performs similarly to other assignment strategies, especially JSQ.

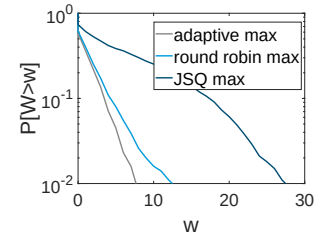
We explore the impact of heterogeneity on the adaptive policy with respect to subsystem utilization by considering $N = 2$ subsystems and fixing the mean



(a) Empirical CCDF of maximum waiting time when epoch length $\Delta = 5$



(b) Empirical CCDF of maximum waiting time when epoch length $\Delta = 10$



(c) Empirical CCDF of maximum waiting time when epoch length $\Delta = 20$

Figure 3.7: Impact of the epoch length Δ on the worst-case waiting time for a system with two subsystems with mean service rates $\mu = [1, 2]$. Our policy tends to outperform uniformly whereas JSQ shows deteriorating performance with increasing Δ. Parameters: arrival rate $\lambda = 1.5$, violation probability $\epsilon = 10^{-4}$.

service rate of subsystem 1 at $\mu_1 = 1$ and varying $\mu_2 \in [1.5, 16]$ at the same time. The average waiting time is then plotted in Fig. 3.5a and 3.5b, which reveals the inclination of the adaptive policy towards picking the faster subsystem. This choice becomes more apparent with increasing heterogeneity.

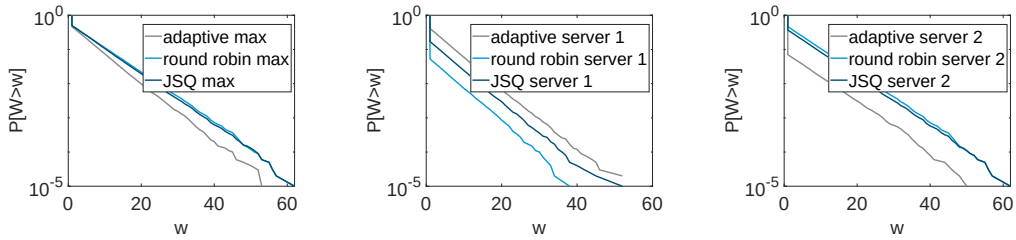
The previous two scenarios are combined in Fig. 3.6a and 3.6b, where we increase the number of subsystems N, each having a service rate $\mu_i = i$. We see that the differences between the assignment strategies shrink with overprovisioning. However, significant differences between the assignment strategies are observed in tightly provisioned systems.

Finally, we focus on the sensitiveness of the assignment policy. This is captured by varying the length of the decision epoch Δ and looking at the corresponding CCDFs of waiting times. In Fig. 3.7, the CCDF of the maximum waiting time is shown along with the subsystem-specific waiting times for a system with $N = 2$ subsystems having arrival rate $\lambda = 1.5$ and heterogeneous service rates $\mu = [1, 2]$. By varying the epoch length in this heterogeneous scenario, we observe that the adaptive policy yields superior performance to RR and JSQ. This performance enhancement can be attributed to the fact that, unlike RR and JSQ, *the statistical characteristics of the arrival and service processes* are taken into account by the adaptive policy as defined in the assignment decision in (3.3.3).

3.4.2 *Markov Modulated On Off Service*

Next, we consider the case when the subsystems in Fig. 3.1 have two channels with memory[2], which we model through service processes S_i that are given as MMOO

2 Recall Fig. 3.2 for a graphical representation of service in such channels.



(a) Empirical CCDF of maximum waiting time

(b) Empirical CCDF of waiting time at the slower subsystem

(c) Empirical CCDF of waiting time at the faster subsystem

Figure 3.8: A system with two MMOO subsystems having *on* state service rates $r = [5, 50]$ and arrival rate $\lambda = 0.5$. The adaptive assignment policy achieves lower worst-case waiting times.

processes with service increment rates at the *On* state $r = [5, 50]$. The underlying transition matrix at the first subsystem is given by $[0.8, 0.2; 0.25, 0.75]$, and at the second subsystem by $[0.85, 0.15; 0.2, 0.8]$. Corresponding initial probabilities of being in the *On* state are 0.5 and 0.3, respectively. Note that the arrivals are still assumed to follow a Poisson distribution for the sake of simplicity. We fix the arrival rate $\lambda = 0.5$. Further, the service envelope is dynamically estimated at the beginning of each decision epoch by inferring the transition matrices through service feedback from respective subsystems as described in Sect. 3.3.2.1. We plot the performance at both subsystems along with the worst-case in Fig. 3.8 through the CCDF of waiting times. As seen, the adaptive system provides a higher decay rate of the worst-case waiting times.

3.5 SUMMARY

In this chapter, we proposed a scheduling policy for TBSs that provides an abstraction for multi-interface communication systems. Although the system has access to multiple interfaces, it uses a single technology to communicate at a time. The state-aware policy makes a transition every decision epoch by choosing the most suitable subsystem. The choice is made by looking at the present workloads of the subsystems and their service guarantees together with the statistical characteristics of the data arrivals. We also illustrated the empirical performance of this adaptive policy for varying subsystem characterizations, for example, for subsystems providing bursty service. The numerical comparison with the established round-robin and JSQ policies revealed that the state-aware policy improved the worst-case system waiting times, especially when the interfaces were heterogeneous. The benefit of the policy can be investigated further by comparing it to a host of other scheduling algorithms such as weighted RR policy.

4

PERFORMANCE OPTIMIZATION VIA EVENT-BASED TRANSITIONS

4.1 INTRODUCTION

To answer our first research question that deals with transitions in parallel systems, we considered time-triggered transitions in Chap. 3. However, in many of today's systems, a transition is followed by events of varying complexity– from temperature rise to traffic congestion (Luthra, Koldehofe, et al. 2018). This motivates us to focus on TBSs making event-based transitions and one such system, a datacenter employing parallelism, was shown on the right of Fig. 1.1. The system, which abstracts TBSs made of a dispatcher[1] having parallel heterogeneous units at its disposal, is illustrated in greater detail in Fig. 4.1. Each heterogeneous unit in the TBS is further made of multiple cores/servers where jobs are scheduled directly by the dispatcher. Thus, our formulation can be adapted to represent any parallel load-balancing system made of heterogeneous groups of workers.

In this TBS, jobs arrive one by one and the dispatcher schedules an incoming job in one of the cores. Put another way, transitions are triggered by the event of job arrival and the scheduling action the dispatcher takes influences the transition of the TBS. In the TBS, cores with similar capabilities form a unit and different units usually have different suitability for executing a certain job. However, cores within a single unit might have different backlogs during a particular job arrival. Thus, the choice of action the dispatcher makes is influenced by the backlogs at the cores and the suitability of the containing unit of a core with regards to the particular job. Note that for systems with a large number of cores/servers, retrieving backlogs from every server imposes an untenable cost on the dispatcher. Hence, the dispatcher only samples a fixed number of servers from each unit and compare their backlogs (Mitzenmacher 2001). The quality of service (QoS) in the TBS can be any user-defined performance metric and to achieve the desired level of performance the user needs to feed in a cost function to the dispatcher that encodes the performance objective. The cost function enables the dispatcher to produce a rank ordering of the sampled servers taking current backlog and unit suitability as input. As our policy, i.e., the choice of action during a transition at a particular system state is shaped by a bespoke cost function, we term it cost-based scheduling (CBS) policy. Our framework helps us track the evolution

1 Also called a scheduler.

of the system state, which further enables the user to evaluate the fitness of the specified cost function to encode the desired performance objective.

The particular example that we take in this chapter is that of a datacenter as shown in Fig. 1.1. The clusters of the datacenter act as heterogeneous units and they consist of finite-buffer servers. The heterogeneity, in this case, might arise due to geographical proximity or hardware capabilities. A user-defined cost function is fed into the dispatcher, which dictates how jobs are scheduled upon every arrival. Consequently, the state evolution is observed, which indicates the efficacy of the cost function to achieve the performance goal.

Scheduling is central to datacenters and analysis of different kind of scheduling policies to achieve certain performance goal has a long history in the queueing theory literature. Some prominent examples include Join the Shortest Queue (JSQ) policy for reduction in the average job response times (Winston 1977), geographical load balancing for reducing energy costs (Neglia, Sereno, et al. 2016) and randomized load balancing to yield similar performance as JSQ without having to incur the cost of monitoring queue lengths at every server (Mitzenmacher 2001). A CBS policy can be seen as a natural extension of the randomized load balancing policies that are inspired by JSQ (Mukhopadhyay, Karthik, et al. 2016) where the flexibility of measuring cluster affinity with regards to jobs is introduced.

State evolution under a cost-based scheduling policy: As mentioned, a CBS policy in the present TBS can be thought of as an extension of randomized load balancing in a supermarket model. We opt for the bespoke framework of the supermarket model instead of the general MDP framework due to its tractability in the context of this particular CBS. In a standard supermarket model, there are M clusters of servers. A stream of jobs arrives at the scheduler, which then routes each job to a server in one of M clusters. The choice of the server is based on randomly sampled queue lengths from each of the M clusters. A popular example of such a policy is the randomized version of the classical JSQ, where the scheduler inspects the queue lengths at d servers, sampled uniformly at random and subsequently dispatches the job to the server having the shortest queue length. To describe the state evolution under a CBS policy in a supermarket model, we first observe that it is sufficient to track the tail distribution of server backlogs for each cluster. Recall that a transition is triggered by the event of job arrival. Upon the arrival of a job, the system takes an action, i.e., assigns a server for serving the job, which leads to a change in the state of the system. We also describe the state, action and policy for the TBS under the technical assumptions in Sect. 4.3.

Further, we see that a CBS policy has the advantage of expressing complex performance objectives. For example, if the TBS wants to follow a JSQ policy when the server backlogs are relatively low and switch to uniform load-balancing for higher backlogs, it can use a cost function that precisely encodes the intended objective; see choices of cost functions in Sect. 4.2 for details. Thus, when

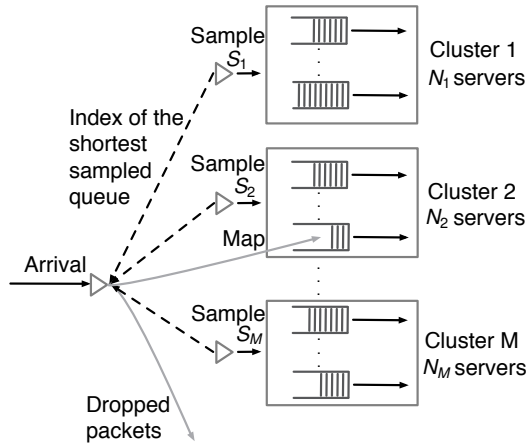


Figure 4.1: Setup and workflow of a CBS policy in the datacenter example introduced in Fig. 1.1. The system consists of M heterogeneous clusters. Cluster i contains N_i homogeneous servers, each requires an exponentially distributed service time with rate μ_i. Upon the arrival of a job, S_i servers are sampled with replacement from cluster i and the one with the shortest queue length is chosen from this cluster. The dispatcher compares the cost of the chosen servers from each cluster and assigns the job to the one with the minimum cost. The job is dropped if the buffer is full.

the cost function is a hybrid of simple cost functions representing elementary scheduling policies, the CBS policy also facilitates a broader transition in terms of the constituent policies. Such transition is usually triggered by the movement of the system state, i.e., the tail distribution of server backlogs from one region to another which is indicated in the hybrid cost function.

We define the performance of the TBS in this case as the evolution of the state over a finite horizon, which also helps compare different CBS policies. We emphasize that instead of optimizing with respect to a long-term accumulated reward, the TBS here adopts a policy that minimizes a user-defined cost at each step. Given the corresponding policy, an analytical estimate of the performance metric of interest is derived and the accuracy and the usefulness of the estimate are discussed subsequently. While analyzing the performance of the TBS under the supermarket model, we realize that an exact analysis becomes tedious for systems with a large number of servers (Singh, Ong, et al. 2015). To that end, we adopt a mean-field framework that enables us to reasonably approximate large system behaviour in constant time.

Contributions: In addition to generalizing the state-of-the-art to incorporate a larger class of policies with complex objectives, we base our analysis on the realistic assumption that the servers have finite buffers. In contrast to recent works

that deal with infinite buffer systems (Mukhopadhyay, Karthik, et al. 2016), our result is of more practical relevance. This is because it has been shown that the behaviour of finite-buffer queues is fundamentally different than their idealized counterparts (Ciucu, Poloczek, et al. 2019a). Moreover, this assumption helps us investigate the effect of finite buffer lengths on certain QoS metrics.

To summarize the contributions, we provide (i) a scaling limit in the form of law of large numbers (LLN) for the empirical tail distribution of server backlogs/queue lengths for an increasing number of servers ($N \to \infty$) for a given CBS policy. (ii) We numerically illustrate that certain QoS metrics decay rapidly with the buffer size, indicating reduced accuracy of results from infinite buffer systems in practice. (iii) We present several numerical results demonstrating the accuracy of the scaling limit and compare the tail distribution of queue length for different cost functions. Note that the techniques used here to derive the scaling limit (weak convergence) are borrowed from (Ethier and Kurtz 1986). Such techniques have been used widely in the context of JSQ-type scheduling strategies (see J. B. Martin and Suhov 1999; Mukhopadhyay and Mazumdar 2016; Mukhopadhyay, Karthik, et al. 2016). We, however, introduce the notion of a cost function that generalized the previous results to non-JSQ type scheduling policies.

The chapter is structured as follows: in Sect. 4.2, we formalize our framework using a queuing set-up for the datacenter example and the CBS policy. Subsequently, a scaling limit is provided in Sect. 4.3. Numerical evaluations are presented in Sect. 4.4 and we summarize our findings in Sect. 4.5.

4.2 SCHEDULING IN LARGE CLUSTERS WITH FINITE BUFFERS

In this section, we first describe the queuing setup and subsequently explain the CBS policy. Our setup is the standard supermarket model (see Mukhopadhyay, Karthik, et al. 2016) with the following modifications: (i) we assume buffers are finite and (ii) we generalize the scheduling algorithm to account for user-defined cost functions, which takes the queue lengths of a random selection of servers as inputs. The second generalization lets us accommodate a wide range of load balancing algorithms and the schemes 1 and 2 in (Mukhopadhyay, Karthik, et al. 2016) can be seen as special cases. We refer the reader to Sect. 6 of (KhudaBukhsh, Kar, et al. 2020b) for an extension to the case of batch arrivals with variable batch sizes.

Queueing theoretic formulation: Let us consider an N-server parallel processor sharing queuing system where the servers are partitioned into M heterogeneous clusters. We assume that $M \ll N$ and that the servers within each cluster are identical (see Fig. 4.1). This structure may arise for reasons such as geographic colocation or equivalent hardware capability. The service discipline is assumed to be FCFS. Recall that $[N] := \{1, 2, \ldots, N\}$. We assume that the i-th cluster contains N_i identical servers having exponential service times with rate parameter μ_i.

Naturally, $N_1 + N_2 + \cdots + N_M = N$ and M is assumed to be fixed. Let I_i be the set of indices of the servers in the i-th cluster. Then, $| I_i | = N_i$ and $\{I_1, I_2, \ldots, I_M\}$ is a partition of $[N]$. That is, $[N] = \bigcup_{i \in [M]} I_i$ and I_i's are disjoint. We fix the buffer size of *all* N servers to be K. This assumption can be relaxed merely at the expense of notational simplicity.

The job arrivals take place according to a Poisson process with rate λ_N. The interarrival and the service times are *all* assumed to be independent of each other. Upon arrival, the dispatcher assigns the job to exactly one server according to a CBS policy, which we describe next.

Cost-based scheduling: We assume that each cluster is equipped with a local router. As seen in Figure 4.1, upon arrival of a job, the main dispatcher sends requests to the M local routers. The local router in the i-th cluster selects S_i servers uniformly at random with replacement. We denote the queue lengths of the sampled servers from the i-th cluster by $Q_{i_1}, Q_{i_2}, \ldots, Q_{i_l}$ where $l = S_i$. The local router computes the queue length of the server with the shortest queue length and its index as

$$\iota_i := \underset{k \in \{i_1, i_2, \ldots, i_l\}}{\arg\min} \{Q_k\}, \forall\, i \in [M] .$$

The index ι_i and the corresponding queue length is then returned to the dispatcher. A tie is broken uniformly at random. Equipped with the indices $\iota_1, \iota_2, \ldots . \iota_M$ from all M clusters, the dispatcher assigns a cost to the corresponding queue lengths. After comparing the resulting costs, the dispatcher assigns the job to the server with the minimum cost. Thus, the job is finally assigned to the server with the following index:

$$\iota := \arg\min_{\iota_k} \{\phi_k(Q_{\iota_k}) \mid k \in [M]\},$$

where ϕ_k denotes the cost function for the k-th cluster. The cost functions are assumed to be user-defined and continuous[2]. As usual, a tie is broken uniformly at random. We further assume that the scheduling task is instantaneous and the jobs leave the system as soon as the service is completed. We now provide some concrete examples of the CBS policy.

Choice of cost functions: CBS can be seen as a generalization of existing load balancing policies. For example, setting $M = 1, S_1 = N, \phi_1(x) = x$ leads to the usual JSQ policy. Similarly, $M = 1, S_1 = 2, \phi_1(x) = x$ corresponds to the power of 2 type JSQ while setting $S_1 = d$ yields the SQ(d) policy. Further, setting $\phi_k(x) = x$ corresponds to the randomized JSQ policy over clusters. Note that any strictly

2 Note that continuity is vacuously satisfied since queue lengths are integer-valued. However, continuity is required if we generalize the cost function to other domains.

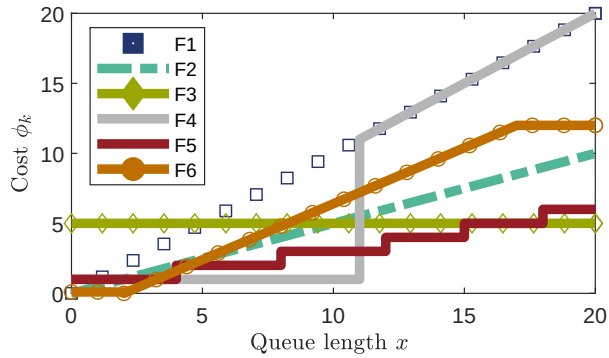


Figure 4.2: Comparison of different cost functions: the function F1 ($x \mapsto x$) represents a JSQ-type policy. In contrast, F2 given by $x \mapsto x/\mu$ (depicted for $\mu > 1$) accounts for the cluster-specific service rate. The policy F3, plotted as $x \mapsto 5$ here, does uniform load balancing. The function F4 implements uniform load balancing for smaller queue lengths and a JSQ-type behaviour beyond a threshold. F5 yields a locally uniform JSQ policy in the sense that it does uniform load balancing when the queue lengths being compared are relatively closer and a JSQ-type behaviour for disparate queue lengths. Finally, F6 shows a cost structure that exhibits JSQ-type behaviour between two threshold queue lengths and enforces uniform load balancing otherwise.

increasing ϕ_k would lead to a JSQ-type policy (or a randomized version of it) whenever ϕ_k's are identical across the M clusters. For example, $\phi_k(x) = x/\mu_k$ yields scheme 2 in (Mukhopadhyay, Karthik, et al. 2016); see F2 in Fig. 4.2.

Apart from the different variants of JSQ, a wide range of cost functions can be designed to achieve other performance objectives as shown in Fig. 4.2. Note that a uniform load balancing across all clusters can be represented by the cost function $c_k = c$ for all k, whereas a constant $\phi_k(x) = c_k$ for all x leads to a preference policy (see F3 in Fig. 4.2). Further, the function F4 corresponds to a policy that does uniform load balancing for small queue length but changes to JSQ as it grows. The policy derived from the cost function F5 can be seen as a locally uniform hybrid JSQ policy, which does uniform load balancing for neighbouring queue lengths but implements JSQ scheduling for disparate queue lengths.

Additionally, combinations of cost functions can be used to represent application semantics on top of the server clusters. One such example is using clusters with cost function F1 in combination with clusters employing F3. Here, the primary load-bearing server clusters have cost function F1 while secondary peak-load absorbing ones use F3. We note that one major advantage of this formulation is the flexibility of choosing different cost functions for different clusters. Finally, some typical real-world examples for the cost functions seen in Fig. 4.2 include mul-

tipath TCP default scheduler for F2 and elastic load balancer in AWS for F3. With these examples in mind, we go on to provide a scaling limit in the next section.

4.3 A SCALING LIMIT

In this section, we derive a scaling limit of the system under the CBS policies as the number of servers N grows to infinity while the proportion of servers in the i-th cluster N_i/N converges to a positive quantity for all $i \in [M]$.

Usefulness of the scaling limit: The most straightforward way to evaluate the performance of CBS policies, where the performance metric can be described by a Markov chain, is to compute the marginal probabilities of the Markov chain by solving the Kolmogorov forward equations. However, the number of equations to solve under this approach increases exponentially with the number of servers N, which renders the method infeasible in most cases[3]. Another alternative is to estimate the probabilities through Monte Carlo simulation, which is also time-consuming for large N.

In contrast, the approach of deriving a mean-field limit for the performance metric, i.e., the process describing the limiting mean of the proportions of servers with certain numbers of unfinished jobs, scales well with the number of servers N. It requires solving a system of $(K+1)M$ ODEs[4], which is a constant of N, as against computational costs that increase with N under previous approaches. This scalable approach further allows us to evaluate the performance and thereby lets us compare the suitability of different CBS policies. In this process, we also demonstrate the empirical accuracy of the limit.

Technical assumptions: We first define the key stochastic processes that let us track the system evolution under CBS policies. Since CBS policies are essentially queue-aware, it is analytically convenient to formulate stochastic processes that keep track of the queue length at each server or summary statistics thereof. Needless to say that the summary statistics are more efficient from a computational point of view. In particular, we choose to work with empirical tail distributions of the queue lengths in each cluster, which turns out to be Markov as we see next.

To keep track of the proportions of servers having no less than a certain number of unfinished jobs in each cluster, we define the stochastic process

$$Z_N(t) := \{Z_{n,i}^{(N)}(t) \mid i \in [M], n = 0, 1, 2 \ldots, K\} . \tag{4.3.1}$$

3 For problems with a particular structure, this approach might still be feasible.

4 Recall that M denotes the number of clusters whereas K denotes the buffer size of each server.

Here $Z_{n,i}^{(N)}(t)$ is an occupancy measure that denotes the proportion of servers in the i-th cluster which have at least n unfinished jobs at time t. Formally,

$$Z_{n,i}^{(N)}(t) := \frac{1}{N_i} \sum_{k \in I_i} \mathbb{1}_{Q_k(t) \geq n} \,. \tag{4.3.2}$$

The script N in Z processes emphasizes their dependence on the total number of servers N. Note that Z_N is a Markov process on $\times_{i \in [M]} \mathcal{Z}_i$ due to exponential arrival and service times and the fact that the future occupancy measure is independent of its past values given the current realization. Here, $\times_{i \in [M]} \mathcal{Z}_i$ denotes the cartesian product of $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_M$, where

$$\mathcal{Z}_i := \{\{a_n\}_{n=0,1,2\ldots,K} \mid a_0 = 1, a_n \geq a_{n+1}, N_i a_n \in \mathbb{N}\}, \quad \forall i \in [M] \,.$$

Specifically, the first coordinate $Z_{0,i}$ denotes the proportion of servers in the i-th cluster having at least zero unfinished jobs, and hence, is 1 for all $i \in [M]$. Further, the proportions $Z_{n,i}$'s are non-increasing in n for a any fixed i. This is because the $N_i a_{n+1}$ servers with at least $(n+1)$ jobs are a subset of the $N_i a_n$ servers having at least n jobs pending. Finally, the condition $N_i a_n \in \mathbb{N}$ is necessary to ensure that a_n is indeed a valid realization of $Z_{n,i}^{(N)}$. Next, we describe how the arrival rate and the number of servers within each cluster scale with the total number of servers N.

A1 **(Arrival rate)** To avoid trivialities, we assume that the arrival rate grows linearly with N, i.e.,

$$\lim_{N \to \infty} \frac{\lambda_N}{N} = \lambda \in \mathbb{R}_+ \,.$$

Note that if λ_N / N diverges, every server will always be full. However, if it converges to zero, the servers will remain idle.

A2 **(Proportion of servers per cluster)** We further assume that asymptotically, each cluster contains a non-vanishing proportion of the total number of servers, which implies a linear growth of cluster sizes with respect to N:

$$\lim_{N \to \infty} \frac{N_i}{N} = \nu_i \in (0,1), \quad \forall i \in [M] \,.$$

Note that this assumption ensures *all* clusters are large enough.

A3 **(Decidability)** This assumption lets us compare different servers across clusters in the context of the relevant CBS policy. More specifically, this implies that given the CBS policy assigns a job to a server with n unfinished jobs in the i-th cluster, we can calculate the minimum possible values of the

sampled queue lengths in different clusters. For example, if the relevant policy is a simple randomized JSQ, the sampled queue lengths for all other clusters must be at least n, i.e., the queue length at the chosen server. That is, given the index ι and the corresponding queue length of the server picked, the cost functions ϕ_i's lets us express the minimum of sampled queue lengths for each cluster. Let $\iota \in I_i$, i.e., the chosen server belongs to the i-th cluster. We define

$$\theta_j(i,x) := \underset{y\in\{0,1,2,\ldots,K\}}{\arg\min} \{\phi_j(y) \geq \phi_i(x)\},$$

for all $i,j \in [M]$, $x \in \{0,1,2,\ldots,K\}$, as the minimum of the sampled queue lengths from the j-th cluster when the relevant CBS policy sends an incoming job to a server in the i-th cluster which has x unfinished jobs. To summarize, A3 ensures $\theta_j(i,x) \neq 0$ for *at least* one $j \neq i$, for all $i \in [M]$ and for all $x > 0$.

Note that the assumption A3 is imposed only to avoid trivialities such as $\theta_j(i,x) = 0$ for all $x \in \{0,1,2,\ldots,K\}$. It is not crucial for the derivations in this chapter. For example, F3 in Fig. 4.2 violates A3.

To put the above formulation into the context of the framework introduced in Chap. 1, we see that the system state S_m at the m-th job arrival is given by $\mathsf{S}_m = Z_N(t)$ if the m-th arrival occurs at time t. Further, the context $\xi_m = \lambda_N$, the rate of job arrival. The action Λ_m here denotes the choice of server implying that the action set $\mathcal{A} = [N]$. Further, the cost of the transition is given by the cost function of the CBS policy itself, i.e., $R(\mathsf{X}_m, \Lambda_m) = \phi_i(x)$ where i denotes the cluster index of the Λ_m-th server and x denotes its queue length. The performance of the TBS is described by the evolution of its state and we look at minimizing the immediate cost R instead of looking at long-term accumulated cost. As mentioned, we perform a bespoke (mean-field) analysis of the system where we look at the time evolution of the empirical tail distribution of server backlogs in the system.

4.3.1 *Main Results*

Having described the technical assumptions, we focus on the main result, which deals with the convergence of the Z_N process to a deterministic function in any bounded interval $\mathcal{T} := [0,\tau]$, $\tau > 0$. We do not include the proofs here; instead we refer the reader to (KhudaBukhsh, Kar, et al. 2020a) for the details of the intermediate steps.

Since $Z_N(t)$ is a proportion, its convergence to a deterministic quantity might seem intuitively reasonable, due to LLN. We adopt the well-known operator semigroup approach to establish convergence of the Markov process Z_N here. We first define a sequence of one-parameter families of operators $\{T_N(t)\}_{t\in\mathcal{T}}$ as

$$T_N(t)f(z_0) := \mathsf{E}[f(Z_N(t)) \mid Z_N(0) = z_0],$$

for all $t \in \mathcal{T}, z_0 \in \times_{i\in[M]}\mathcal{Z}_i$ and for all continuous functions $f : \times_{i\in[M]}\mathcal{Z}_i \to \mathbb{R}$. The family $\{T_N(t)\}_{t\in\mathcal{T}}$ defines a strongly continuous (contraction) semigroup (Engel and Nagel 1999; Ito and Kappel 2002) due to the Chapman-Kolmogorov property of Markov processes (Ethier and Kurtz 1986, Chapter 4).

Now in light of (Ethier and Kurtz 1986, Chapter 4, Theorem 2.11, p. 176), convergence of $T_N(t)$ to a limiting operator semigroup T implies convergence of the Markov process Z_N. Further, following standard technique, we establish convergence of the operator semigroup T_N by showing that the corresponding sequence of generators A_N converges to the generator A of T. Same approach has been adopted in (Mukhopadhyay, Karthik, et al. 2016) for the infinite-buffer case. To summarize, we take the following steps to establish convergence of Z_N in Thm. 4.3.1 below: 1) we first prove convergence of the generators A_N to A, which we state in Lemma 4.3.1; 2) next we show that T_N converges to T, which we mention as Thm. 4.3.2; and finally 3) convergence of Z_N to its scaling limit z follows by (Ethier and Kurtz 1986, Thm. 2.11).

Note that in order to state Thm. 4.3.1, we need to define the space where the z process belongs. We first recall that Z_N lies in the space $\times_{i\in[M]}\mathcal{Z}_i$ and each $\mathcal{Z}_i$ is finite as the corresponding proportions can only assume finitely many values. However, as $N \to \infty$, the proportions can converge to any real number in $[0, 1]$. This motivates us to define the space for the limiting occupancy measure for a single server as

$$\mathcal{Z} := \{\{a_n\}_{n=0,1,\ldots,K} \mid a_0 = 1, a_n \geq a_{n+1}, a_n \in [0, 1]\} .$$

Thus, we expect that the state space of the scaling limit of Z_N will be $\mathcal{Z}^M$, the M-fold cartesian product of $\mathcal{Z}$.

Theorem 4.3.1 (Convergence of the server cluster proportions)**.** *If for some non-random $z_0 \in \mathcal{Z}^M$, $Z_N(0)$ converges to z_0 in probability, we have*

$$Z_N \xRightarrow{\mathcal{D}} z, \text{ as } N \to \infty , \tag{4.3.3}$$

where the limiting process z satisfies

$$z(t) = z(0) + \int_0^t \mathbb{F}(z(s))\mathrm{d}s , \tag{4.3.4}$$

with $z(0) = z_0$, and the function $\mathbb{F}(u) := \{\mathbb{F}_{n,i}(u) \mid i \in [M], n = 0, 1, 2, \ldots, K\}$ is defined as

$$\mathbb{F}_{0,i}(u) := 0, \quad \forall i \in [M] ,$$

$$\mathbb{F}_{n,i}(u) := \frac{\lambda}{\nu_i}\left((u_{n-1,i})^{S_i} - (u_{n,i})^{S_i}\right) \prod_{j\in[M]\setminus\{i\}} (u_{\theta_j(i,n-1),j})^{S_j} - \mu_i\left(u_{n,i} - u_{n+1,i}\right) ,$$

for $i \in [M]$, $n = 1, 2, \ldots, K$, and $u \in \mathcal{Z}^M$. For a definition of weak convergence, see (Ethier and Kurtz 1986, Chapter 3, p. 107), (Billingsley 1999).

Intuitively, the operator $\mathbb{F}(z)$ can be thought of as the derivative of the deterministic process z. Note that the finiteness of the buffers implies the system is always stable. We now concretely state the results used in showing the intermediate steps of Thm. 4.3.1, which are proven in (KhudaBukhsh, Kar, et al. 2020a).

Lemma 4.3.1 (Convergence of the generators)**.** *Let $\mathcal{C}$ denote the space of all real-valued continuous functions defined on $\mathcal{Z}^M$, i.e., $\mathcal{C} := \mathcal{C}(\mathcal{Z}^M)$. We consider the subspace $\mathcal{C}_D \subseteq \mathcal{C}$ for which the partial derivatives*

$$\frac{\partial}{\partial u_{n,i}} z(t,u), \quad \frac{\partial^2}{\partial u_{n,i}^2} z(t,u), \quad \text{and} \quad \frac{\partial^2}{\partial u_{n,j} u_{n,i}} z(t,u)$$

exist $\forall u \in \mathcal{Z}^M$ and are uniformly bounded. Then, $\forall f \in \mathcal{C}_D$,

$$\lim_{N\to\infty} A_N f(u) = \frac{\mathrm{d}}{\mathrm{d}t} f(z(t,u))\Big|_{t=0}, \tag{4.3.5}$$

where z is the solution of (4.3.4).

Theorem 4.3.2 (Convergence of the operator semigroup)**.** *For any $f \in \mathcal{C}_D$, and $t \in \mathcal{T}$, the following convergence of the operator semigroup $\{T_N(t)\}_{t\in\mathcal{T}}$ holds for any CBS policy:*

$$\lim_{N\to\infty} \sup_{u \in \times_{i\in[M]} \mathcal{Z}_i} |T_N(t) f(u) - T(t) f(u)| = 0 . \tag{4.3.6}$$

Here the limiting operator semigroup $\{T(t)\}_{t\in\mathcal{T}}$ is defined by $T(t)f(u) := f(z(t,u))$, having the corresponding generator

$$Af(u) := \lim_{h\to 0+} \frac{T(t+h)f(u) - T(t)f(u)}{h} .$$

This is equivalent to $\frac{\mathrm{d}}{\mathrm{d}t} f(z(t,u))\Big|_{t=0}$ where z is the solution of the integral equation (4.3.4).

Properties of the scaling limit: Next, we discuss some properties of the scaling limit. As the limit is given by a system of ODEs (4.3.4), it is natural to ask if the system has a unique solution, which turns out to be the case. Like previous results, the derivations of these properties can be found in (KhudaBukhsh, Kar, et al. 2020a).

Lemma 4.3.2. *Irrespective of the starting point $u \in \mathcal{Z}^M$, the solution to the integral equation (4.3.4) is unique on the finite time interval $\mathcal{T} = [0, \tau]$.*

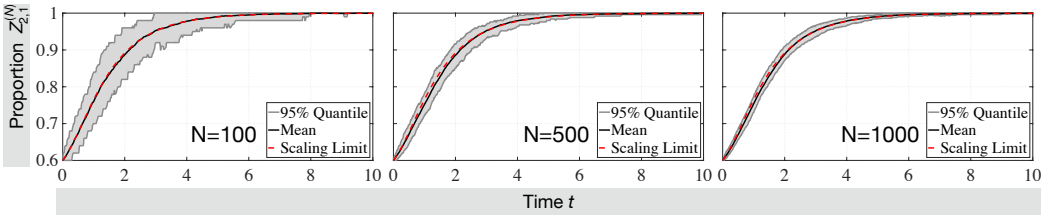


Figure 4.3: Accuracy of the limit from Thm. 4.3.1: the time evolution of $Z_{2,1}^{(N)}$ from (4.3.2) (empirical proportions) for increasing number of servers N vis-a-vis the scaling limit $z_{2,1}$, i.e., the proportion of servers in cluster 1 having at least 2 jobs in the queue. The fluctuation of $Z_{2,1}^{(N)}$ around $z_{2,1}$ diminishes with increasing N.

The next result states that the solution of (4.3.4) is smooth with respect to the initial conditions, which is crucial for establishing the convergence of generators in Lemma 4.3.1. Here we denote $z(0) = u$ and rewrite the limiting process $z(t)$ as $z(t,u)$ to indicate the dependence of the process on the initial value u.

Lemma 4.3.3. *For all $u \in \mathcal{Z}^M$, the partial derivatives*

$$\frac{\partial}{\partial u_{n,i}} z(t,u), \quad \frac{\partial^2}{\partial u_{n,i}^2} z(t,u), \quad \text{and} \quad \frac{\partial^2}{\partial u_{n,j} u_{n,i}} z(t,u)$$

exist and are uniformly bounded above

$$\begin{aligned} \left|\frac{\partial}{\partial u_{n,i}} z(t,u)\right| &\leq \exp(at)\,, \\ \left|\frac{\partial^2}{\partial u_{n,i}^2} z(t,u)\right|\,, \ \left|\frac{\partial^2}{\partial u_{n,j} u_{n,i}} z(t,u)\right| &\leq \exp(bt)\,, \end{aligned} \tag{4.3.7}$$

for some $a, b \in \mathbb{R}_+$.

Next, we numerically evaluate the results stated in this section.

4.4 NUMERICAL EVALUATIONS

In this section, we focus on systems employing a CBS policy and look at their evolution with time, steady-state behaviour and also the approximate effect of buffer length on job loss. Recall that Thm. 4.3.1 lets us circumvent the tedious process of simulating the cluster system to gain insights into its performance, especially when each cluster grows large. For the sake of completeness, we first numerically validate this limit where we compare the empirical queue length proportion with the theoretical limit in Fig. 4.3. We consider three simulation scenarios with increasing number of servers and use 10^2 Monte Carlo simulations for

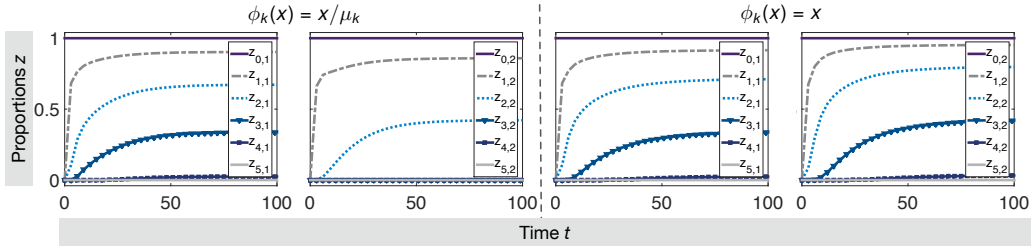


Figure 4.4: Comparison of the evolution of cluster queue length proportions over time for cost functions ϕ: $\phi_k(x) = x/\mu_k$ (left half) and $\phi_k(x) = x$ (right half). Simulation parameters: $M = 2$, $K = 5$, $\lambda = 1/4$, $(\nu_1, \nu_2) = (0.4, 0.6)$, $(\mu_1, \mu_2) = (1/2, 1/4)$, time horizon $= [0, 100)$. At a given time point, $z_{j,i}$ denotes the proportion of servers in cluster i having queue length greater than or equal to j. As expected, the CBS policy under $\phi_k(x) = x/\mu_k$ picks servers from the faster cluster ($\mu_1 = 1/2$) more often.

each run to obtain the corresponding mean/percentiles. Unless stated otherwise, the general numerical settings for the plots are as follows: time horizon $= [0, 100)$, number of clusters $M = 2$, server buffer size $K = 5$, arrival rate $\lambda = 1/4$, proportion of servers per cluster[5] $(\nu_1, \nu_2) = (0.4, 0.6)$, service rates within the clusters $(\mu_1, \mu_2) = (1/2, 1/4)$.

In Fig. 4.3, we see that the scaling limit evolves in close proximity with the empirical mean of the Monte Carlo simulations and the variance of the empirical proportion diminishes with a growing number of servers N. We emphasize that the scaling limit is used here to describe the behaviour of the system in the transient phase.

Next, we plot the time evolution of the proportions of servers in each cluster for certain queue lengths in Fig. 4.4. Here, the cost functions $\phi_k(x) = x$, $\phi_k(x) = x/\mu_k$ represent randomized JSQ and service-rate-weighted randomized JSQ scheduling, respectively. Observe that the service-rate-weighted randomized JSQ takes advantage of the faster server leading to comparatively larger proportions of longer queue lengths.

In Fig. 4.5, we explore the evolution of a system under different CBS policies in heterogeneous environments. For both clusters, we consider cost functions F1 (randomized JSQ), F3 (uniform load balancing) and F4 (initially uniform and changing to randomized JSQ beyond a fixed queue length) from Fig. 4.2. Here we change the simulation parameters to $K = 8$, $\lambda = 1/8$, and $(\mu_1, \mu_2) = (1, 1/16)$ to underscore heterogeneity. Note that the cost function F3 and F4 use almost all servers in the slower cluster eventually. In comparison, fewer servers remain idle under F1(left subplot). Also, in the right subplot of Fig. 4.5 we see that certain

5 Recall assumption A2 that limitting proportion of servers per cluster should be positive.

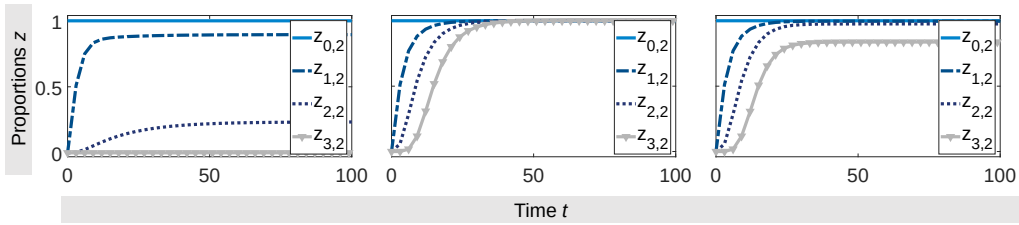


Figure 4.5: Queue length proportions $z_{j,2}$s at the slower of two heterogeneous clusters. Cost functions chosen from left to right (see Fig. 4.2): F1 (randomized JSQ), F3 (uniform load balancing) and F4 (initially uniform and changing to randomized JSQ at a threshold queue length of 4). Recall that $z_{j,2}$ is the proportion of servers in the slower cluster having no less than j jobs. F1 is seen to have shorter queues (left) while F3 does not allow idling of servers (middle). In contrast to F3, intermediate queue lengths are more probable under F4.

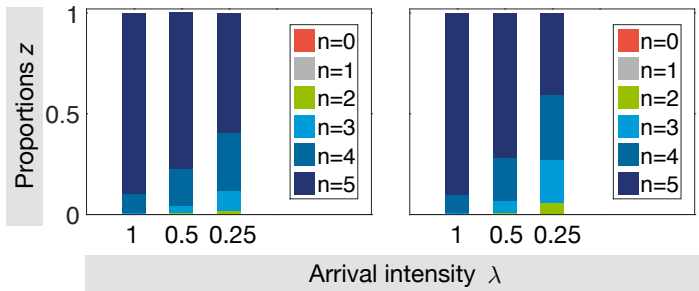


Figure 4.6: Steady-state queue length (n) proportions for the first (left) and the second (right) cluster for varying arrival intensities. We use the cost function $\phi_k(x) = x/\mu_k$ where x and μ_k denote the queue length and the cluster service rate, respectively. The disparity in occupancy measure between clusters tends to diminish with growing arrival rate.

servers have more workload under F4. This is explained by the fact that until a threshold queue length (3 in case of Fig. 4.5), F4 employs uniform load balancing. We also see that compared to F3, F4 results in a higher proportion of servers having intermediate queue length.

In Fig. 4.6, we look at cluster-wise steady-state proportions of queue lengths under *high*, *medium* and *low* arrival rates, i.e., $\lambda \in \{1/4, 1/2, 1\}$. Recall that we use server buffer size $K = 5$ and service rates within the clusters $(\mu_1, \mu_2) = (1/2, 1/4)$. Under the low arrival regime, the observed occupancy measure indicates moderately higher load on the faster cluster. However, the disparity tends to die out as the arrival rate increases.

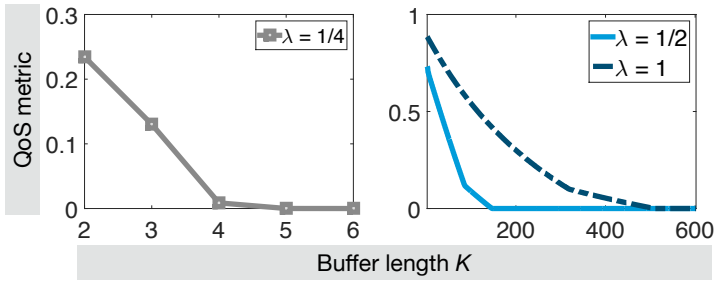


Figure 4.7: Effect of the buffer size on the QoS metric time-averaged proportion of full servers for different arrival rates: the buffer length is seen to have a diminishing impact on the QoS metric.

Finally, we show the effect of the buffer size on the time-averaged proportion of server bottleneck (i.e., full servers) in the whole system in Fig. 4.7. Note that this QoS metric may serve as an indicator of job loss. For the simulation, we use a long time horizon $= [0, 400)$ to compute the time-averaged proportion. We see that the buffer length has a rapidly diminishing impact on the QoS metric, which presents a favourable argument for smaller buffers from the perspective of system design. We emphasize that such empirical insight cannot be gained from the corresponding infinite buffer results such as (Mukhopadhyay, Karthik, et al. 2016).

4.5 SUMMARY

In this chapter, we considered a queueing model that abstracts a parallel system with heterogeneous groups of servers. The dispatcher schedules jobs by looking at server backlogs and the capabilities of the groups with regards to the present job. This model provides us with a framework to analyze a large class of TBSs employing parallelism and execute transition based on certain events. Specifically, we considered the scenario of a datacenter made of heterogeneous clusters of finite-buffer servers. To track the performance of the system, we chose a certain occupancy measure of server backlogs as the metric and observed its evolution.

The state evolution of the system was shaped by a certain scheduling policy that allowed incorporating different QoS objectives compared to the state of the art that primarily focuses on a singular quality metric. We observed that CBS policies generalized the framework of common JSQ policy and the popular power of two or SQ(d) policies could be obtained as special cases.

To evaluate the performance of CBS policies, we adopted the mean-field approach. Note that under a Markovian setup, performance evaluation can be alternatively done by computing the marginal probabilities of the system at different time points. This approach requires either solving the Kolmogorov equations

or estimation via Monte Carlo simulations. However, both procedures become computationally infeasible as the number of servers grows.

On the contrary, a scaling limit for the empirical tail distribution of server backlogs could be calculated in constant time. The limit, characterized by a system of ODEs, was further used: (i) to derive estimates of QoS, such as buffer filling proportions, given a CBS policy; and (ii) to compare performance under different CBS policies. In our evaluations, we also showed how certain metrics of interest such as queue length proportions and full server ratios can be obtained numerically from the scaling limit.

5 THROUGHPUT OPTIMIZATION

5.1 INTRODUCTION

So far in this thesis, we have considered transitions in open parallel systems, which were triggered by time or events. In this chapter, we focus on our second research question which deals with transitions in *closed* batch-processing systems and study their impact on the resulting performance. To that end, we consider closed transition-based systems (TBSs) where transitions happen only upon system reconfiguration. The considered TBS serves jobs in batches to distribute operational overhead, which leads to shorter service time per job. Service batching has been known to enhance performance and is routinely used in certain communication systems (Cree 2018; Wen, Yang, et al. 2016). Although higher batch size yields shorter per job service time, the considered TBS has multiple servers and increasing the batch size eventually leads to under-utilization. This presents a non-trivial performance optimization problem and we focus on methods to calculate the optimal batch size that leads to optimal system throughput.

We model the TBS as a closed queueing system with n clients and m parallel servers where each client switches between *active* and *inactive* state. In the active state, it submits a job for processing and while inactive, it awaits the execution of the job. Consequently, a client can have at most one job in the system as it produces a new job only after the previously submitted one is served. Further, clients wait for a random time after getting the response to submit a new job. Each server processes jobs in batches of size $k \geq 1$. Therefore, as long as there are less than k newly submitted jobs, a batch waits to be formed. Once formed, batches are forwarded for processing. Evidently, batches wait in a common queue if all servers are busy. We show a schematic description of the TBS in Fig. 5.1[1]. The TBS represents a large class of real-world data-processing systems such as databases employing Multi Query Optimization (Rehrmann, Binnig, et al. 2018; Sellis 1988; Thomson, Diamond, et al. 2012).

In our analysis, we consider TBSs serving single as well as multiple job types. Note that for the multiple batch type case, batches of different types not only have different mean service times, but also a certain service priority order among

1 All times are assumed to be exponentially distributed with respective rate parameters. The batching and the service rate depend on the batch size k. We will show the validity of this convenient assumption by fitting model parameters from traces of a real-world system.

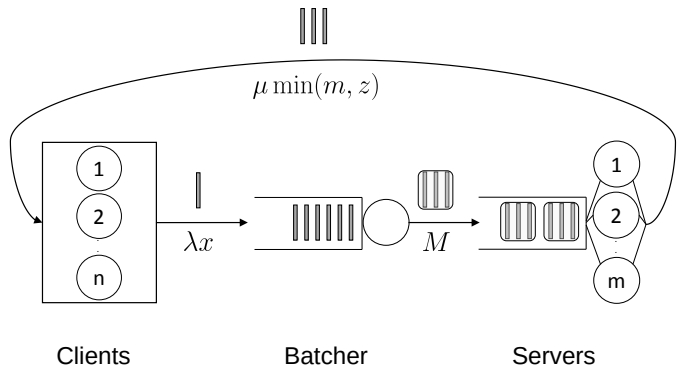


Figure 5.1: The closed batching system with n clients and m servers: clients can be either in active or inactive state. With x active clients, the effective job generation rate is λx. When there are y available jobs, the batcher merges them to produce batches of size k at rate $M \lfloor y/k \rfloor$. The service station contains m parallel servers with a common queue. Service rate of each server is μ, implying an overall *batch* service rate of $\mu \min(m, z)$ when z batches are available.

them. For example, `write` jobs usually have non-preemptive priority over `read` jobs in databases to produce up-to-date results.

Existing literature on queueing systems with batch arrivals and batch service disciplines (Bailey 1954; Bolch, Greiner, et al. 2005; Chaudhry and Templeton 1983; Deb 1978; Henderson and Taylor 1990) mostly focuses on open queueing systems or different properties of interest in closed systems such as the product form[2]. However, the considered TBS cannot be reduced to any of the proposed frameworks and consequently, we adopt the model in Fig. 5.1. Using the introduced queuing system, we first provide an exact analysis, which solves for the balance equations in a Markov chain. The analysis requires at least $\omega\left(n^4\right)$ computational time where n denotes the number of clients. To deal with the computational complexity and the fact that many of today's batch-processing systems support a large number of clients, we adopt a mean-field approach, which yields optimal (asymptotic) throughput in $O(1)$ time.

To illustrate the usefulness of our results, we run experiments in a prototype of a commercial database system. In our experiments, a job refers to a query such as an SQL string executing `read` or `write` operations. Batching in this context involves merging queries of similar nature into a SQL string, whose processing time depends on factors like the query type and the size of the batch. In systems

2 For a more thorough discussion, see Chap. 2.

serving jobs of multiple types, the processing times are naturally different and leads to different speedup functions.

We organize the remainder of the chapter as follows. We first describe the queueing model and the corresponding optimization problem. Subsequently, we provide the mean-field model and the asymptotic result in Sect. 5.3. We discuss the model for two job types in Sect. 5.4, which we further generalize in Sect. 5.7.3, although with certain restrictions. We then present numerical and experimental evaluations in Sect. 6.5 before summarizing our contributions.

5.2 QUEUEING MODEL AND OPTIMIZATION GOAL

In this chapter, we model our TBS using a closed queueing system consisting of three stations: job producer, job batcher, and service station. Jobs in the system are routed sequentially along these stations. The producer station has n clients. In the beginning, each client is assigned a token, which enables them to submit a new job. Upon job submission, the token is seized and the job is routed to the job batcher. The batcher merges jobs to create a single job of size k at rate $M(k)$, once k jobs are available. Thereupon, the batch is forwarded to the service station, which comprises of m servers. The merged job is then compiled, executed, and the result is split and sent back to the respective clients, all of which is considered as processing of the batch. Each server performs the processing of *batches* at rate $\mu(k)$ and in a first come first server (FCFS) order. In the end, together with the response, each client receives its token back, which lets it submit a new job. Observe that the rate at which a new job is sent to the batcher depends on the number of clients with a token, referred to as *active* clients in the following, rather than the total number of clients. Furthermore, releasing a token only upon receiving the response ensures that the total number of jobs in the system remains the same as the number of clients n. For a schematic illustration of the job flow in the system, see Fig. 5.1.

Recall that the batching of jobs is motivated by the fact that it is faster to serve jobs together, i.e., in batches, than to serve them sequentially. This implies that the gain from batching grows with increasing batch size, a phenomenon commonly referred to as *speedup*. However, this gain is not unmitigated. For example, the batching step introduces additional overhead. Further, having a batch size beyond a certain threshold, where the number of batches in the system becomes less than the number of servers, leads to the idling of the available servers. This in turn affects the throughput of the system, i.e., the number of jobs served at the service station per unit time. As higher batch sizes may cause servers to become idle, we end up with a non-trivial performance tradeoff. Naturally, we seek to find the optimal batch size k^*, which maximizes the system throughput. To this end, we first model the closed queueing system as a continuous time Markov chain (CTMC)

to derive its steady-state behaviour. Further, as the optimal batch size k^* depends on the system parameters, we recalculate it whenever the system is reconfigured.

We assume that the time taken by each client to produce a new job is exponentially distributed with rate λ. Thus, for x active clients (i.e., clients with a token), the producer station produces a job and submits it to the batching station at rate λx. Let us further denote the number of jobs at the batching station by y and the number of *batches* at the service station by z. Therefore, the triple (x, y, zk) uniquely represent the state of the system and the state space is given by

$$\mathcal{S} = \left\{(x_1, x_2, x_3) \in \mathbb{Z}_+^3 : x_1 + x_2 + x_3 = n, k | x_3\right\} .$$

Although the third component of (x, y, zk) is redundant, we retain the triple representation due to its immediate interpretation. We see that the system evolves as a continuous time Markov chain and the jump rates are given by

$$\begin{aligned} (x, y, zk) &\xrightarrow{\lambda x} (x-1, y+1, zk) \ , \ x > 0 \\ &\xrightarrow{M(k)\lfloor y/k \rfloor} (x, y-k, (z+1)k) \ , \ y \geq k \\ &\xrightarrow{\mu(k)\min(m,z)} (x+k, y, (z-1)k) \ , \ z > 0 \ . \end{aligned} \tag{5.2.1}$$

Note that (5.2.1) refers to the fact that in the state (x, y, zk), either one job can proceed from the producer station to the batching station with rate λx as there are x active clients, *or* k jobs can leave the batching station and move to the server with rate $M(k)\lfloor y/k \rfloor$, *or* k more clients receive their token back at rate $\min(m, z)\mu(k)$ making them active again. The jump rates to other states are zero.

The system attains a steady-state and the unique steady-state distribution $\boldsymbol{\pi}_0$ is given by the solution of the equation $\boldsymbol{\pi} \cdot \mathbf{Q} = 0$. The existence and uniqueness is guaranteed by the fact that the chain is irreducible and finite and hence positive recurrent; see Sect. 5.7.1 for a detailed argument. Here, $\mathbf{Q}$ denotes the intensity matrix of the chain, i.e., $\mathbf{Q}(r, s)$ is the jump rate from state r to s where $r, s \in \mathcal{S}$. Due to the state-dependent non-linear rates, the solution $\boldsymbol{\pi}_0$ is obtained numerically rather than in closed form. Using the steady state distribution $\boldsymbol{\pi}_0$, we can compute the steady state system throughput as

$$\Theta(k) := \sum_{(x,y,zk)\in\mathcal{S}} \boldsymbol{\pi}_0(x, y, zk)\, k\mu(k) \min(m, z) \ , \tag{5.2.2}$$

which gives the optimal batch size

$$k^* := \arg\max_{k\in\mathcal{K}} \Theta(k) \ . \tag{5.2.3}$$

Here $\mathcal{K} = \{1, 2, 3, \ldots, K\}$, where K is the maximum admissible batch size in the underlying system. We note the fact that finding the solution of (5.2.3) runs in

$\omega\left(n^4\right)$ time as it requires solving $\boldsymbol{\pi} \cdot \mathbf{Q} = 0$ for every $1 \leq k \leq K$ in (5.2.2), and for a particular batch size k, the size of $\mathbf{Q}$ is $O\left(\frac{n^2}{k}\right)$.

5.3 MEAN-FIELD MODEL

Given the size of modern data-processing systems, the Markovian approach followed in Sect. 5.2 becomes computationally infeasible even for moderately sized systems. This motivates us to adopt a mean-field approach where the number of servers m is scaled with the number of clients n. Further, we assume that the batching step is instantaneous. That is, when there are $(k-1)$ jobs in the batching station, the arrival of a new job leads to immediate batch formation and the number of jobs in the batching station jumps to 0. This assumption is motivated by empirical observations; for example, in the commercial database system used for evaluation experiments, the batching step is about 50 times faster than the service step. As an additional benefit, this assumption leads to further simplification in our analysis.

Let us denote by $X^{(n)}(t)$ the number of active clients in the system at time $t \geq 0$. Therefore, the number of jobs/queries in the system during this time is $n - X^{(n)}(t)$. Observe that $(X^{(n)}(t), t \geq 0)$ is a Markov process on $\{0, 1, \ldots, n\}$ with the following jump rates:

$$q^{(n)}(x \to x-1) = \lambda x \,,$$
$$q^{(n)}(x \to x+k) = \mu(k) \min\left(m, \left\lfloor \frac{n-x}{k} \right\rfloor\right) \,.$$

Here $x \in \{0, 1, \ldots, n\}$ and $q(i \to j)$ is the jump rate from state i to state j. The Markov process $(X^{(n)}(t), t \geq 0)$ attains a unique steady-state distribution because it is irreducible and the state space is finite. However, it is difficult to solve the matrix equation $\pi^{(n)} Q^{(n)} = 0$ analytically due to the non-linear state-dependent rates, as mentioned in Sect. 5.2. Instead, we derive a bound on the system throughput as follows. Under the stationary distribution, the following equality holds:

$$\lambda \mathsf{E}[X] = k\mu(k) \mathsf{E}\left[\min\left(m, \left\lfloor \frac{n-X}{k} \right\rfloor\right)\right] \,. \tag{5.3.4}$$

Using Jensen's inequality, we obtain

$$\lambda \mathsf{E}[X] \leq k\mu(k) \min\left(m, \frac{n - \mathsf{E}[X]}{k}\right) \,.$$

Note that the system throughput $\Theta^{(n)}$ is given by the RHS of (5.3.4), which implies the following bound on the throughput :

$$\mathsf{E}\left[\Theta^{(n)}\right] \leq \min\left(k\mu(k)m, \frac{n\lambda\mu(k)}{\lambda + \mu(k)}\right) \,. \tag{5.3.5}$$

Although $\Theta^{(n)}$ is a function of the batch size k, we have dropped this dependency for brevity. Next, we establish that the tightness of the bound in the asymptotic regime, i.e., as $n, m \to \infty$ with $m = \alpha n$ and $\alpha \in \mathbb{R}_+$.

Let us consider the process $(w^{(n)}(t), t \geq 0)$, which denotes the proportion of active clients in the system, i.e.,

$$w^{(n)}(t) := X^{(n)}(t)/n \ .$$

The process $(w^{(n)}(t), t \geq 0)$ is a *density dependent jump Markov process* (Kurtz 1970) with rates

$$\begin{aligned} q^{(n)}(w \to w - 1/n) &= n\lambda w \\ q^{(n)}(w \to w + k/n) &= n\mu(k) \min\left(\alpha, \frac{1}{n}\left\lfloor \frac{n - nw}{k} \right\rfloor\right) \ , \end{aligned}$$

where $w := x/n$. With this definition, we are ready to state the main result of this chapter:

Theorem 1. (i) Let $w^{(n)}(0) \to w_0 \in [0,1]$ as $n \to \infty$ in probability. Then

$$\sup_{0 \leq t \leq T} \|w^{(n)}(t) - w(t)\| \to 0$$

in probability as $n \to \infty$, where $(w(t), t \geq 0)$ is the unique solution of the ODE:

$$\dot{w}(t) = f(w(t)) \ , \quad w(0) = w_0 \ ,$$

and $f : [0,1] \to \mathbb{R}$ is defined as

$$f(w) = k\mu(k) \min\left(\alpha, \frac{1-w}{k}\right) - \lambda w \ .$$

(ii) For all $w_0 \in [0,1]$, $w(t) \to w^*$ as $t \to \infty$, where w^* is the unique solution of $f(w) = 0$, i.e.,

$$w^* = \min\left(\frac{\mu(k)}{\lambda + \mu(k)}, \frac{\alpha k \mu(k)}{\lambda}\right) \ . \tag{5.3.6}$$

(iii) The sequence of stationary measures of the process $(w^{(n)}(t), t \geq 0)$, denoted by $\pi_w^{(n)}$, converges weakly to δ_{w^*} as $n \to \infty$.

Proof. For part (i), note that the limiting expected drift of the process $(w^{(n)}(t), t \geq 0)$ given $w^{(n)}(t) = w$ converges uniformly to the continuous function f, i.e., for any $w \in [0,1]$,

$$\lim_{n\to\infty} \lim_{h \to 0} \frac{1}{h} \mathsf{E}\left[w^{(n)}(t+h) - w^{(n)}(t) | w^{(n)}(t) = w\right] = f(w) \ .$$

Further, observe that $f : [0,1] \to \mathbb{R}$ is Lipschitz continuous by the following facts: (1) linear functions are Lipschitz continuous, (2) for Lipschitz continuous functions F and G, $cF + dG$ is Lipschitz continuous for $c, d \in \mathbb{R}$, (3) $|F|$ is Lipschitz continuous if F is Lipschitz continuous, and (4) $\min(F, G) = \frac{F+G}{2} - \frac{|F-G|}{2}$. Part (i) thus follows from Theorem 3.1 of (Kurtz 1970).

To show part (ii), we first see that the unique solution to the equation $f(w) = 0$ is given by (5.3.6). Without loss of generality, let us assume that $w_0 \geq w^*$. Thus $w(t) \geq w^*$ for all $t \geq 0$ as $w(t)$ is continuous and $\dot{w}(t) = 0$ for $w(t) = w^*$. We now define the distance function $\phi(t) = w(t) - w^*$. Evidently, $\phi(t) \geq 0$ for all $t \geq 0$. Further,

$$\begin{aligned}
\dot{\phi}(t) = \dot{w}(t) &= f(w) \\
&= f(w) - f(w^*) \\
&= -\lambda(w - w^*) + \\
&\quad k\mu(k) \left[\min\left(\alpha, \frac{1 - w(t)}{k}\right) - \min\left(\alpha, \frac{1 - w^*}{k}\right) \right] \\
&\leq -\lambda\phi \,,
\end{aligned}$$

where the last inequality follows from the fact that $w(t) \geq w^*$ for all $t \geq 0$. Therefore, $\phi(t) \leq \phi(0)e^{-\lambda t}$, which implies $w(t) \to w^*$.

For part (iii), observe that the stationary measure $\pi_w^{(n)}$ is tight as it is defined on the compact space $[0,1]$. Thus, part (iii) follows from Theorem 2 of (Bortolussi and Gast 2016). □

A direct consequence of the theorem is:

$$\lim_{n\to\infty} \lim_{t\to\infty} \mathsf{E}\left[w^{(n)}(t)\right] = \lim_{t\to\infty} \lim_{n\to\infty} \mathsf{E}\left[w^{(n)}(t)\right] = w^* \,.$$

Put another way, we have derived the limiting value of the normalized throughput as:

$$\lim_{n\to\infty} \mathsf{E}\left[\Theta^{(n)}/n\right] = \lambda w^* \,,$$

establishing the asymptotic tightness of the bound from (5.3.5).

To obtain optimal asymptotic throughput for given system parameters, we maximize the corresponding proportion of active clients w^* with respect to the batch size k. Note that according to (5.3.5), the asymptotically optimal batch size can be obtained by solving the following optimization equation:

$$k^* = \max_k \min\left(\frac{\mu(k)}{\lambda + \mu(k)}, \frac{\alpha k \mu(k)}{\lambda}\right) \,. \tag{5.3.7}$$

For non-increasing $\mu(k)$ and non-decreasing $k\mu(k)$, the optimal solution k^* can be obtained by solving the following equation

$$\frac{\mu(k)}{\lambda + \mu(k)} = \frac{\alpha k \mu(k)}{\lambda} \,. \tag{5.3.8}$$

Thus, the problem of finding the optimal batch size k^* can be reduced to the simpler problem of solving a polynomial equation whenever $\mu(k)$ assumes a polynomial form. The solution k^* can be used to approximate the optimal batch size for finite systems as long as n and m are large. The principal advantage of this approach is that (5.3.8) can be solved in time independent of the system size n. Thus, we can use (5.3.7) to calculate k^* every time the system is reconfigured and set it as the batch size while running the system. We will see in Sect. 6.5 that, apart from reducing the computation time, (5.3.7) yields reasonable approximation even in practical settings.

5.4 THE TWO JOB-TYPE CASE

In this section, we consider a system with two types of jobs, e.g., `write` and `read` in a database system. Each type benefits from batching and the corresponding speedup function can possibly assume different forms. Note that we only consider the case where jobs of the same type can be batched together, typically true for database systems. In addition, we assume that there is a preemptive priority in service scheduling between the two types. This is typical for a database system where `write` jobs are prioritized over `read` jobs, although in a non-preemptive manner. However, we will see that empirically the preemptive model fits quite well to the practical non-preemptive regime. Next, we describe our approaches to calculate the optimal batch size in this case.

5.4.1 *Queueing Model and Exact Solution*

Recall that the client station has n clients, each producing a job at rate λ when it is active (i.e., with a token) and we denote the number of active clients by x. Further, a job produced by an active client can be of type 1 with probability p or type 2 with probability $(1-p)$. We assume, without loss of generality, that the first type is prioritized over the second type in the service station. We further denote the number of type 1 and type 2 jobs in the batching station by y_1 and y_2, respectively. Out of these jobs, the batching station merges k_i jobs of type i into a batch at rate $M_i(k_i)\,\lfloor y_i/k_i \rfloor$, whenever $y_i \geq k_i$, $i \in \{1,2\}$. A batch is then forwarded to the service station immediately. The service station consists of m parallel servers and serves the batches in FCFS order with *preemptive priority* given to the batches of the first type.

Let us denote the total number of batches of the first type by z_1. Preemptive priority implies that out of them, v_1 many occupy one server each where $v_1 = \min(m, z_1)$. The rest of the servers can possibly be occupied by batches of the second type. The quadruple (x, y_1, y_2, z_1k_1) describes the state of the system uniquely and the corresponding state space is given by:

$$\mathcal{S} = \left\{ (x_1, x_2, x_3, x_4) :\in \mathbb{Z}_+^4 : x_1 + x_2 + x_3 + x_4 \leq n, k | x_4 \right\}.$$

This is due to the fact that the number of type 2 jobs which are already batched is

$$z_2k_2 = (n - x - y_1 - y_2 - z_1k_1) \ ,$$

out of which v_2k_2 are actually in service and the rest are queued in the service station, where

$$v_2 = \min(\max(0, m - z_1), z_2) \ . \tag{5.4.9}$$

The system evolves as a continuous time Markov chain and the corresponding jump rates are given by:

$$\begin{aligned}
s &\xrightarrow{\lambda x p} s - \mathbf{e_1} + \mathbf{e_2} \ , \ x > 0 \\
&\xrightarrow{\lambda x(1-p)} s - \mathbf{e_1} + \mathbf{e_3} \ , \ x > 0 \\
&\xrightarrow{M_1(k_1)\lfloor y_1/k_1 \rfloor} s - k_1\mathbf{e_2} + k_1\mathbf{e_4} \ , \ y_1 \geq k_1 \\
&\xrightarrow{v_1\mu_1(k_1)} s + k_1\mathbf{e_1} - k_1\mathbf{e_4} \ , \ z_1 \geq 1 \\
&\xrightarrow{v_2\mu_2(k_2)} s + k_2\mathbf{e_1}, \ v_2 \geq 1 \ .
\end{aligned} \tag{5.4.10}$$

Here we have used the shorthand $s = (x, y_1, y_2, z_1k_1)$ and e_j is the unit vector of befitting size having the j-th component as unity. The jump rate to any other state is zero.

By arguments similar to Sect. 5.2, the chain is irreducible and positive recurrent. Hence, once we populate the intensity matrix $\mathbf{Q}$ using (5.4.10), we can derive the steady state distribution $\boldsymbol{\pi}_0$ by solving $\boldsymbol{\pi} \cdot \mathbf{Q} = 0$. While it is possible to optimize jointly for k_1 and k_2, a uniform batch size across job type is generally favoured in batching systems (see, e.g., Giannikis, Alonso, et al. 2012; Rehrmann, Binnig, et al. 2018; Sellis 1988). Denoting $k := k_1 = k_2$, the steady state throughput can be expressed as:

$$\Theta_p(k) = \sum_{s \in \mathcal{S}} \boldsymbol{\pi}_0(s) k(\mu_1(k)v_1 + \mu_2(k)v_2) \ , \tag{5.4.11}$$

where $s = (x, y_1, y_2, z_1 k)$, $v_1 = \min(m, z_1)$, and v_2 is derived in (5.4.9). Therefore, the optimal batch size is given by:

$$k^* = \arg\max_{k \in \mathcal{K}} \Theta_p(k) \ , \tag{5.4.12}$$

where $\mathcal{K} = \{1, 2, 3, \ldots, K\}$ and K denotes the maximum possible batch size in the underlying system.

5.4.2 *Mean-field Formulation: Preemptive Priority*

In this section, we discuss the two job-type case with preemptive priority in the context of the mean-field formulation where we additionally assume that the batching step is instantaneous[3]. The system in this case can be uniquely described by the number of active clients and the number of type 1 jobs. This is because there cannot be more than $(k_1 - 1)$ jobs of type 1 that are yet to form a batch; implying that the number of unbatched jobs of type 1 is $mod(x_2, k_1)$, where x_2 denotes the number of type 1 jobs in the system. Analogously, we can derive the number of type 2 jobs which are yet to be batched assuming work conserving behaviour of the server. From now on, we will use the notation μ_1 and μ_2 instead of $\mu_1(k_1)$ and $\mu_2(k_2)$ when the context is clear.

Let us denote by $X_1^{(n)}(t)$ and $X_2^{(n)}(t)$ the numbers of active clients and the number of type 1 jobs in the system at time $t \geq 0$, respectively. Therefore, at time t, the number of type 2 jobs is $n - X_1^{(n)}(t) - X_2^{(n)}(t)$, the number of type 1 batches in service is $\left(\min\left(m, \left\lfloor X_2^{(n)}(t)/k_1 \right\rfloor\right)\right)$, and the number of type 2 batches in service is

$$\begin{aligned} &\min\left(m - \min\left(m, \left\lfloor X_2^{(n)}(t)/k_1 \right\rfloor\right), \left\lfloor (n - X_1^{(n)}(t) - X_2^{(n)}(t))/k_2 \right\rfloor\right) \ , \quad \text{i.e.,} \\ &\min\left(\max\left(0, m - \left\lfloor X_2^{(n)}(t)/k_1 \right\rfloor\right), \left\lfloor (n - X_1^{(n)}(t) - X_2^{(n)}(t))/k_2 \right\rfloor\right) \ . \end{aligned}$$

Note that $(X_1^{(n)}(t), X_2^{(n)}(t), t \geq 0)$ is a Markov process on the state space

$$\mathcal{S} = \left\{(x_1, x_2) \in \mathbb{Z}_+^2 : x_1 + x_2 \leq n\right\}$$

with the following jump rates:

3 Recall that we assumed the same in Sect. 5.3.

$$
\begin{aligned}
&q((x_1,x_2)\to(x_1-1,x_2+1)) = \lambda p x_1\\
&q((x_1,x_2)\to(x_1-1,x_2)) = \lambda(1-p)x_1\\
&q((x_1,x_2)\to(x_1+k_1,x_2-k_1)) = \mu_1 \min\left(m, \left\lfloor \frac{x_2}{k_1}\right\rfloor\right)\\
&q((x_1,x_2)\to(x_1+k_1,x_2-k_1)) = \mu_1 \min\left(m, \left\lfloor \frac{x_2}{k_1}\right\rfloor\right)\\
&q((x_1,x_2)\to(x_1+k_2,x_2)) =\\
&\qquad \mu_2 \min\left(\max\left(0, m-\left\lfloor \frac{x_2}{k_1}\right\rfloor\right), \left\lfloor \frac{n-x_1-x_2}{k_2}\right\rfloor\right)
\end{aligned}
$$

Like Sect. 5.3, we consider the scaled process $w^{(n)}(t) = (w_1^{(n)}(t), w_2^{(n)}(t))$ where $w_i^{(n)}(t) = X_i^{(n)}(t)/n$, $i \in \{1,2\}$ and prove a similar result.

Theorem 2. (i) Assume $w^{(n)}(0) \to w_0 \in [0,1]^2$ as $n \to \infty$ in probability. Then

$$\sup_{0\le t\le T} \|w^{(n)}(t) - w(t)\| \to 0$$

in probability as $n \to \infty$, where $(w(t) = (w_1(t), w_2(t)), t \ge 0)$ is the unique solution of the system of ODEs given by:

$$\dot{w}_1(t) = f_1(w(t))\,, \quad \dot{w}_2(t) = f_2(w(t))\,, \quad w(0) = w_0\,,$$

where $f = (f_1, f_2) : [0,1]^2 \to \mathbb{R}^2$ is defined as

$$
\begin{aligned}
f_1(w) &= -\lambda w_1 + k_1\mu_1 \min\left(\alpha, \frac{w_2}{k_1}\right) +\\
&\qquad k_2\mu_2 \min\left(\max\left(0, \alpha - \frac{w_2}{k_1}\right), \frac{1-w_1-w_2}{k_2}\right)\\
f_2(w) &= \lambda p w_1 - k_1\mu_1 \min\left(\alpha, \frac{w_2}{k_1}\right).
\end{aligned}
$$

(ii) For $w_0 \in [0,1]^2$, $w(t) \to w^*$ as $t \to \infty$, where $w^* = (w_1^*, w_2^*)$ is the unique solution of $f(w) = 0$ and is given by:

$$
\begin{aligned}
w_1^* &= \min\left(\frac{\mu_1\mu_2}{\mu_1\lambda(1-p) + \mu_2\lambda p + \mu_1\mu_2}, \frac{k_1k_2\mu_1\mu_2\alpha}{k_1\mu_1\lambda(1-p) + k_2\mu_2\lambda p}\right)\,,\\
w_2^* &= \frac{\lambda p w_1^*}{\mu_1}\,.
\end{aligned}
\tag{5.4.13}
$$

(iii) The sequence of stationary measures of the process $(w^{(n)}(t), t \geq 0)$, denoted by $\pi_w^{(n)}$, converges weakly to δ_{w^*} as $n \to \infty$.

Proof. Proof of part (i) follows by arguments similar to the proof of part (i) of Theorem 1.

For part (ii), we start by observing that w^* is the unique solution of $f(w) = 0$. Next, we show that w^* is globally attractive. Let us first define a linear transform $(w_1, w_2) \to (z_1, z_2)$ where $z_1 = w_1 + w_2$ and $z_2 = w_2$. The transformed system then evolves as:

$$\frac{dz_1}{dt} = \begin{cases} -\lambda(1-p)(z_1 - z_2)\,, & \text{if } z_2 \geq k_1\alpha \\ -\lambda(1-p)(z_1 - z_2) + \mu_2(1 - z_1)\,, & \text{if } \frac{1-z_1}{k_2} + \frac{z_2}{k_1} < \alpha \\ -\lambda(1-p)(z_1 - z_2) - \frac{k_2}{k_1}\mu_2 z_2 + k_2\mu_2\alpha\,, & \text{if } \frac{1-z_1}{k_2} + \frac{z_2}{k_1} \geq \alpha \end{cases}$$

$$\frac{dz_2}{dt} = \begin{cases} \lambda p(z_1 - z_2) - k_1\mu_1\alpha\,, & \text{if } z_2 \geq k_1\alpha \\ \lambda p(z_1 - z_2) - \mu_1 z_2\,, & \text{otherwise.} \end{cases}$$

The stationary point is mapped to (z_1^*, z_2^*), where $z_1^* = \min(z_{11}^*, z_{12}^*)$ and $z_2^* = \eta z_1^*$ with

$$z_{11}^* = \frac{(\mu_1 + \lambda p)\mu_2}{\mu_1\lambda(1-p) + \mu_2\lambda p + \mu_1\mu_2},\ z_{12}^* = \frac{k_1k_2(\mu_1 + \lambda p)\mu_2\alpha}{k_1\mu_1\lambda(1-p) + k_2\mu_2\lambda p},\ \text{and}\ \eta = \frac{\lambda p}{\mu_1 + \lambda p}\,.$$

Observe that the system is a piece-wise linear and we can study stability of each region independently:

Case 1: $k_1\alpha \geq 1$, $k_2\alpha \geq 1$
Since the domain of interest is $1 \geq z_1 \geq z_2 \geq 0$, the system in this case can be described by:

$$\begin{aligned} \frac{dz_1}{dt} &= -\lambda(1-p)(z_1 - z_2) + \mu_2(1 - z_1)\,, \\ \frac{dz_2}{dt} &= \lambda p(z_1 - z_2) - \mu_1 z_2\,. \end{aligned} \tag{5.4.14}$$

Evidently, this system can be represented as a linear dynamical system $\dot{z} = Az + b$; the eigenvalues θ of $A \in \mathbb{R}^{2\times 2}$ satisfy

$$\theta^2 + (\lambda + \mu_1 + \mu_2)\,\theta + c_0 = 0\,,$$

for some constant c_0. We observe that the real parts of the eigenvalues are strictly negative, implying global attraction to the unique stationary point $(z_{11}^*, \eta z_{11}^*)$.

Case 2: $k_1\alpha \geq 1$, $k_2\alpha < 1$
We start by observing that the domain of interest in this case is separated into two regions by the line

$$L_1 : z_2 = \frac{k_1}{k_2}(k_2\alpha - 1 + z_1)\,,$$

where each region has their respective linear equations. Further, we consider the line

$$L_2 : z_2 = \eta z_1 \ .$$

See that $\dot{z}_2(t) \geq 0$ iff z lies below the line L_1. Therefore, when the initial point is below L_1, the state of the system stays there and vice-versa. This implies that the fixed point(s), if exist(s), should lie on L_2. Let us further denote by z_{10} the z_1-coordinate of the intersection point of L_1 and L_2:

$$z_{10} = \frac{k_1(\mu_1 + \lambda p)(1 - k_2\alpha)}{k_1\mu_1 + k_1\lambda p - k_2\lambda p} \ .$$

We first consider the subcase: $z_{11}^* \leq z_{12}^*$. Quick calculations reveal that this leads to $z_{10} \leq z_{11}^* \leq z_{12}^*$. If the initial point of the system is below L_1, it evolves according to (5.4.14). Thus an argument similar to *Case 1* establishes convergence to the fixed point $(z_{11}^*, \eta z_{11}^*)$. In case the system starts from a point above L_1, the evolution in the beginning is given by:

$$\begin{aligned} \frac{dz_1}{dt} &= -\lambda(1-p)(z_1 - z_2) + k_2\mu_2(\alpha - \frac{z_2}{k_1}) \ , \\ \frac{dz_2}{dt} &= \lambda p(z_1 - z_2) - \mu_1 z_2 \ . \end{aligned} \tag{5.4.15}$$

Simple calculations reveal that the real parts of the eigenvalues corresponding to the linear representation are negative. We also see that the fixed point for this system is: $(z_{12}^*, \eta z_{12}^*)$. Therefore, the system crosses L_1 and that point onward the evolution is given by (5.4.14). Looking from the point of view of system convergence, this is equivalent to starting the system below L_1 and in this case the convergence to $(z_{11}^*, \eta z_{11}^*)$ is already established. Thus, when $z_{11}^* \leq z_{12}^*$, the system converges to $(z_{11}^*, \eta z_{11}^*)$ regardless. For $z_{11}^* > z_{12}^*$, we observe that $z_{10} \geq z_{11}^* \geq z_{12}^*$ and by a similar argument, the system converges to $(z_{12}^*, \eta z_{12}^*)$.

Case 3: $k_1\alpha < 1, k_2\alpha < 1$
Let us first consider the case where the system starts at (z_1, z_2) such that $1 \geq z_1 \geq z_2 \geq k_1\alpha$. We first prove that the system eventually reaches a region that satisfies $z_2 \leq k_1\alpha$. This is true because until we have $z_1 \leq k_1\alpha$, the evolution is given by

$$\begin{aligned} \frac{dz_1}{dt} &= -\lambda(1-p)(z_1 - z_2) \ , \\ \frac{dz_2}{dt} &= \lambda p(z_1 - z_2) - k_1\mu_1\alpha \ ; \end{aligned} \tag{5.4.16}$$

which implies z_1 and z_2 decreases indefinitely and hence reaches a point where $z_2(t) \leq k_1\alpha$.

Thus, without loss of generality, we assume the initial point satisfies $z_2 \leq k_1\alpha$. Let us consider the subcase $\eta \leq k_1\alpha$. Like *Case 2*, we see that either $z_{10} \leq z^*_{11} \leq z^*_{12}$ or $z_{10} \geq z^*_{11} \geq z^*_{12}$ and the proof mimics the arguments in *Case 2*.

For the other subcase $\eta > k_1\alpha$, we first show that $z^*_{11} \geq z^*_{12}$, which is equivalent to

$$\mu_1\mu_2k_1k_2\alpha \leq k_1\mu_1\lambda(1-p)(1-k_2\alpha) + k_2\mu_2\lambda p(1-k_1\alpha) \ .$$

Since $\eta > k_1\alpha$, it is enough to show

$$\mu_1\mu_2k_2\eta \leq k_1\mu_1\lambda(1-p)(1-k_2\alpha) + k_2\mu_2\lambda p(1-\eta) \ .$$

The inequality above is equivalent to $k_1(\mu_1 + \lambda p)(1-p)(1-k_2\alpha) \geq 0$. As in *Case 2*, we notice that if the system starts above L_1, it evolves according to (5.4.15) and hence converges to $(z^*_{12}, \eta z^*_{12})$. When it starts below L_1, the evolution in the initial phase is given by (5.4.14) and the system approaches towards $(z^*_{11}, \eta z^*_{11})$. Thus the evolution equation eventually changes to (5.4.15) and the system converges to $(z^*_{12}, \eta z^*_{12})$ regardless.

Case 4: $k_1\alpha < 1, k_2\alpha \geq 1$
Similar to *Case 2*, we consider an initial point satisfying $z_2 \leq k_1\alpha$. Let us first deal with the subcase $\eta \leq k_1\alpha$. Similar to the subcase of *Case 3* where $\eta > k_1\alpha$, we observe this implies $z^*_{11} \leq z^*_{12}$ as we have $k_2\alpha \geq 1$ in this case. The proof of convergence from a starting point below or above L_1 now follows similar arguments.

The remaining subcase is $\eta > k_1\alpha$. As seen in *Case 2*, we notice that either $z_{10} \leq z^*_{11} \leq z^*_{12}$ or $z_{10} \geq z^*_{11} \geq z^*_{12}$ and using identical arguments, we can prove convergence to $(z^*_{11}, \eta z^*_{11})$ or $(z^*_{12}, \eta z^*_{12})$, respectively. This establishes global attraction under every scenario. Note that we have explicitly stated the actual limit as $(z^*_{11}, \eta z^*_{11})$ or $(z^*_{12}, \eta z^*_{12})$ for respective cases.

Finally, part (iii) of the theorem follows from similar arguments used in part (iii) of Thm. 1. □

The above theorem implies that the asymptotic throughput can be expressed as a linear combination of w^*_1 and w^*_2. Thus, with known forms of the speedups $\mu_1(k)$ and $\mu_2(k)$, we can maximize the asymptotic throughput jointly over k_1 and k_2. Clearly, the time required to solve this optimization problem is independent of the system size n and the asymptotic solution estimates the optimal batch size for moderately sized finite systems quite accurately, as we will see in evaluations in Sect. 6.5. Further, we consider a more general case with multiple job types in 5.7.3 where we assume that the batch sizes are equal across types.

5.5 EVALUATION

In this section, we show the accuracy of our model first by running simulations and subsequently by running real database queries in a prototype of a commercial database system.

5.5.1 *Simulation-based Evaluation*

In this section, we compare results from the exact and the mean-field approach with the results we observe in a simulated system. Note that *exact* result indicates the model throughput obtained by solving for the steady-state distribution numerically whereas *simulation* represents the observed throughput when the system is run according to (5.2.1). The unit of time is seconds and consistent with the real system, we assume a linear form of speedup for simulations. Finally, we use the following system parameters for simulating throughput in the single job-type case[4]:

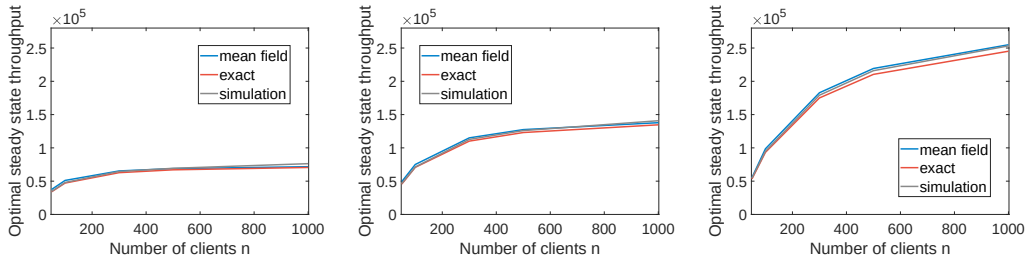
- query generation rate $\lambda = 5 \cdot 10^3$,
- batch service time $1/\mu(k) = 3.6 \cdot 10^{-4} + 5.2 \cdot 10^{-5}\ k$,
- batch formation time $1/M(k) = 7.2 \cdot 10^{-6} + 1 \cdot 10^{-6}\ k$.

For the system with two job types, type 1 is assumed to have higher priority and is generated with 20% probability. We only adjust the service rates leaving other parameters unchanged, i.e., the type 1 service time $1/\mu_1(k) = 1/(5 \cdot \mu_2(k))$ and type 2 service time $1/\mu_2(k) = 5.4 \cdot 10^{-4} + 5.3 \cdot 10^{-4}\ k$.

In Figures 5.2 and 5.3, we study the steady-state throughput for the exact model, the mean-field model and simulations. Note that this is done for both the one job-type case and the two job-type case where the first job type has preemptive priority over the second. In these figures, the number of servers m is varied and the corresponding steady-state throughput is obtained as a function of the number of clients n or of the batch size k. Fig. 5.2 deals with the optimal steady state throughput where it is plotted as a function of the number of clients n, for fixed number of servers m. Observe that both the non-asymptotic/exact model and the mean-field model closely mimic the optimal steady-state throughput seen via simulations. We see a similar trend in Fig. 5.3 where we show the optimal total steady-state throughput when the job of the first type has preemptive priority over the second.

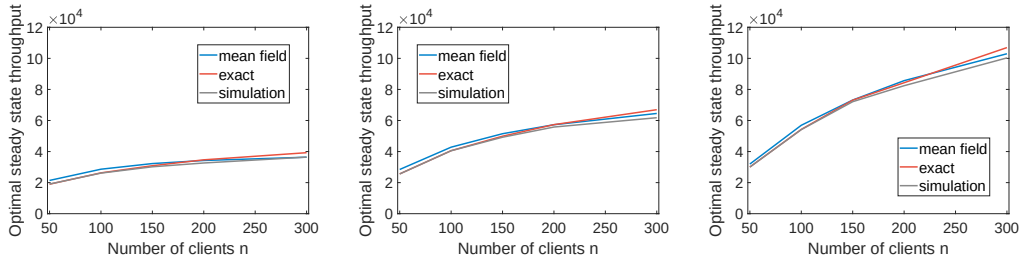
We now look at the behaviour of the steady-state throughput as a function of the batch size k. Fig. 5.4 demonstrates how the exact model and the mean-field model closely capture the steady-state throughput from simulation and

4 These parameters are chosen from the range of values observed in the experiments conducted in the prototype; see Sect. 5.5.2.



(a) $m = 4$ (servers) **(b)** $m = 8$ **(c)** $m = 16$

Figure 5.2: The optimal steady-state throughput for an increasing number of clients n and a fixed value of the number of servers m in the single job-type case with linear speedup. Output from the mean-field model, the non-asymptotic/exact model and simulations tend to agree. The mean-field analysis calculates the optimal throughput as $mk^*\mu(k^*)$, where k^* is given in (5.3.8).



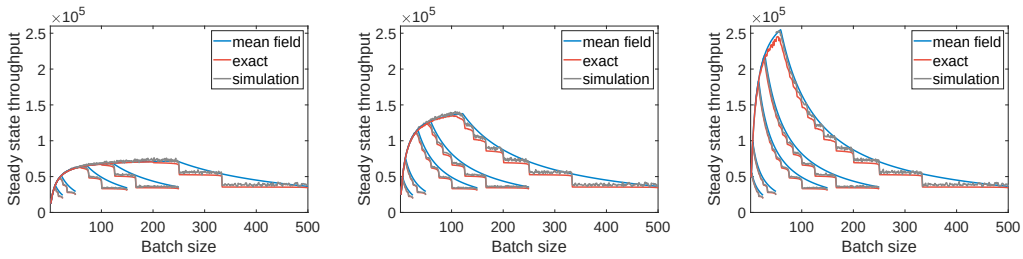
(a) $m = 4$ **(b)** $m = 8$ **(c)** $m = 16$

Figure 5.3: The optimal total steady-state throughput for a system serving two job types: linear speedup and preemptive priority is assumed.

the corresponding optimal batch size k^*. Further, Fig. 5.5 compares the total throughput for the two job-type case. To conclude, we see that throughput as a function of the batch size k, and hence the optimal batch size k^* from both the non-asymptotic/exact and the mean-field model agree with simulation.

5.5.2 *Prototype-based Evaluation*

In this section, we discuss our model performance when the experiments are run in a prototype of a commercial database system. We run two sets of experiments where in the first set we execute only `read` queries and consider both `read` and `write` queries in the latter. Additionally, the `write` queries are generated 20% of the time and have non-preemptive priority over the `reads`. We refer the reader to Sect. 6.2 of (Kar, Rehrmann, et al. 2020) for the details on system description and the data collection procedure.



(a) $m = 4$ **(b)** $m = 8$ **(c)** $m = 16$

Figure 5.4: Steady-state throughput of the system for increasing batch size k and fixed number of servers m; each set of lines in a subplot corresponds to a value of $n \in \{50, 100, 300, 500, 1000\}$ from left to right. Note that the non-asymptotic/exact and simulated throughput takes a sharp downturn at batch sizes where the number of maximum possible active server decreases by one. This becomes more apparent for larger batch sizes as the corresponding relative change is higher. Both the exact and the mean-field model accurately capture the steady-state throughput and the optimal batch size (the batch size corresponding to the peak point).

5.5.2.1 *Fitting the Experimental Data*

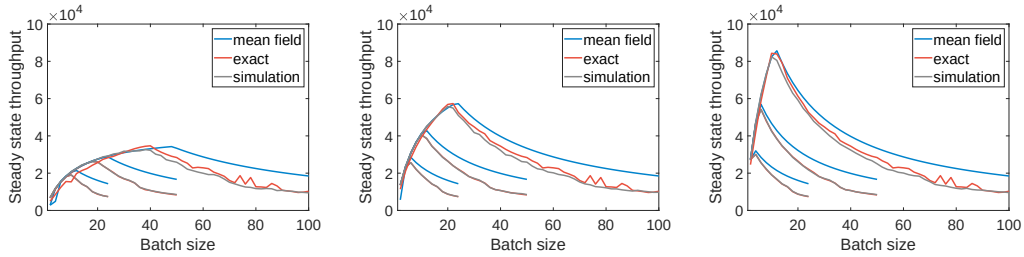
To find estimates of the optimal batch sizes under two modelling approaches, we first estimate the model parameters using the observation from experiments. For estimation of the parameters of the speedup, we use the standard optimal experiment design method. This allows us to efficiently estimate the speedup parameter by running the experiments run for a few selected batch sizes. We only discuss the estimation procedure for the one job-type case as the estimation in the two job-type case follows an identical workflow.

We represent the batching speedup through $g : \mathbb{N} \mapsto \mathbb{R}_+$ where $g(k) = 1/\mu(k)$. Since we focus only on non-trivial speedups, we impose sub-additivity: $g(k_1 + k_2) \leq g(k_1) + g(k_2)$. To fit a speedup function to the observation from the database-prototype, we consider following speedup forms for their tractability:

- $g_1(k) = ak + b$ with $a < 1$
- $g_2(k) = \gamma k^{\alpha}$ with $\alpha < 1$
- $g_3(k) = c \log k + d$ with $c < 1$.

We estimate respective parameters using the mean service times for different batch sizes.

For our estimation procedure, we first calculate a set of batch sizes, which minimizes the estimation error. We take a linear regression-based approach where the speedup function is transformed into a linear combination of weights $\mathbf{w}$ and



(a) $m = 4$ **(b)** $m = 8$ **(c)** $m = 16$

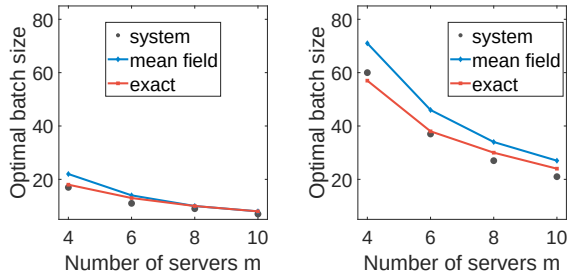
Figure 5.5: Steady state throughput for the two job-type case for a fixed number of server $m \in \{4, 8, 16\}$ under preemptive priority: each set of lines in a subplot corresponds to a value of $n \in \{50, 100, 200\}$ (from left to right).

features $\boldsymbol{\phi}(k)$. Assuming the error of the responses in this formulation, i.e., the mean service times, are normally distributed, the ordinary least square (OLS) estimates of the weights can be calculated by solving the normal equations of a standard linear model. For the experiment design, i.e., choosing the set A, which contains the batch sizes for which the mean service time data is obtained, we use a D-optimal design (Pukelsheim 1993) that minimizes the log-determinant of the covariance matrix of our OLS estimator. Here, the size of the set A is set according to time and cost budget. Let us denote by ψ_k the indicator variable, which takes the value 1 if $k \in A$. Then we can rewrite the problem as an integer optimization problem where we have to pick $|A|$-many ψ_k's that minimizes the log-determinant of the covariance matrix. Recall that $|A|$ is already fixed according to a fixed budget. We solve this optimization problem using the CVX package (Grant, Boyd, et al. 2008) after relaxation of the indicator variables ψ_k; see Chap. 7.5.1 of (Boyd and Vandenberghe 2004) for details.

Denoting the set of sample service times for the batch size $k \in A$ as S_k, we partition S_k into subsets of size 100 and compute a mean service time from each of these subsets. Denoting the average of service times of the j-th subset corresponding to batch size k as Y_{jk}, we find the speedup function g minimizing the corresponding OLS estimation error, i.e., $g = g_m$ where $m = \arg\min_i e_i$ and $e_i = \sum_{k \in A} \sum_j \left(g_i(k|\hat{\boldsymbol{\theta}}_i) - Y_{jk}\right)^2$. Here, $\hat{\boldsymbol{\theta}}_i$ is the OLS estimate of $\boldsymbol{\theta}_i$, i.e., if we denote the parameter space for $\boldsymbol{\theta}_i$ as $\boldsymbol{\Theta}_i$, we have $\hat{\boldsymbol{\theta}}_i = \arg\min_{\theta_i \in \Theta_i} \sum_{k \in A} \sum_j \left(g_i(k) - Y_{jk}\right)^2$.

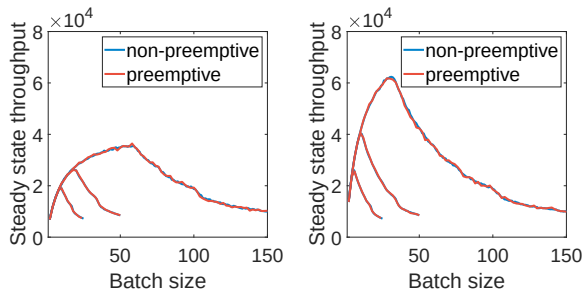
5.5.2.2 *Evaluation*

In order to estimate the speedup function for batch formation and service, we fix a measurement budget $\sim 5\%$, i.e., we observe average service time and batching time for $\sim 5\%$ of all possible batch sizes. Adopting the optimal experimental design approach mentioned in the previous section, we calculate the probing set



(a) $n = 100$ **(b)** $n = 300$

Figure 5.6: Comparison of the observed optimal batch sizes with corresponding model estimates for an increasing number of servers for a fixed number of client $n \in \{100, 300\}$. The experiment here is done only with `read` jobs; the optimal batch size decreases as the number of servers grows.



(a) $m = 4$ **(b)** $m = 8$

Figure 5.7: Equivalence of preemptive and non-preemptive priority in simulation: the steady-state throughput for the two job-type case moves in close proximity. Each set of lines corresponds to a fixed number of server $n \in \{50, 100, 300\}$ (from left to right).

A for $n \in \{100, 300\}$ clients and observe the mean service times and batching times from separate runs. The mean service times and the batching times for batch sizes $k \in A$ then serve as the input data for the estimation procedure.

The estimated service and batching rates let us populate the intensity matrix Q using (5.2.1) and we find the corresponding steady-state distribution numerically. Subsequently, we compute the steady-state throughput using (5.2.2) and find the optimal batch size k^*. The same process is repeated for different number of servers m and for the number of clients $n \in \{100, 300\}$. The database prototype where the experiments are run has at most $m = 10$ servers. For the sake of completeness, we also conduct an exhaustive experiment (i.e., for all possible batch sizes) to derive the empirical optimum for $m = 4$ and $n = 100$. We present our estimates of the optimal batch size from two approaches in Fig. 5.6 for an

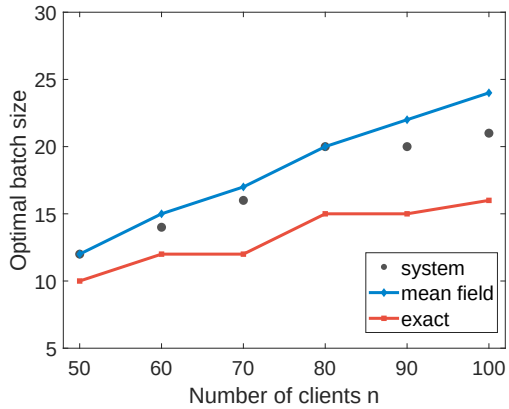


Figure 5.8: Comparison of optimal batch size k^* for `read` and `write` job types: the experiment is done for $m = 4$ servers and number of clients $n \in \{50, 60, 70, 80, 90, 100\}$ under non-preemptive priority of `writes` over `reads`. For the mean field model, optima are estimated by the preemptive model in Sect. 5.4.2 while the non-asymptotic/exact model estimates k^* as per Sect. 5.7.2. Observe that the modelled and the system optima increase in close proximity.

increasing number of servers while fixing the number of clients $n \in \{100, 300\}$. Note that both the exact model and the mean-field model accurately capture the drop in the optimal batch size k^* due to the addition in the number of servers m.

Finally, we run experiments where jobs can either be of type `read` or `write`. We set the generation probability of `read` queries to 80% and that of the `write` queries to 20%. Consistent with real-world databases, the `write` jobs are given priority over the `read` jobs. The system provides a non-preemptive priority to the `write` jobs; although, for the mean-field model we assume preemptive priority. Nonetheless, the difference between two priority setups vanishes in the stationary asymptotic regime, as seen in simulations; see Fig. 5.7. In Fig. 5.8, we stack up the estimated and the actual optimal batch sizes; the estimates from the mean-field model predict the actual optima particularly well.

5.6 SUMMARY

In this chapter, we considered closed batch-processing TBSs where transitions occurred upon system reconfiguration. The batching benefit in such TBSs is expressed in terms of a speedup that formulates the mean service time as a function of batch size. Although increasing the batch size leads to shorter processing time per job, it can potentially cause the idling of servers. Hence, upon system reconfiguration, a batch size is calculated that maximizes the throughput

of the system. The calculated optimum depends upon system parameters and is used until the next reconfiguration/transition. An example of the considered TBS is a standard database system where clients await the response of an input query until it submits a new one.

To calculate the optimal batch size maximizing the throughput, we propose an exact and a mean-field technique. While the former captures the system dynamics accurately, it becomes computationally infeasible for large systems– thus leading us to the mean-field approach. We evaluate both approaches in a prototype of a commercial database system where we observe a close match with empirical system behaviour.

5.7 DEFERRED PROOFS

5.7.1 *Irreducibility of the Closed Queueing System*

Proposition 1. *The Markov chain underlying the queueing system in Sect. 5.2 is irreducible.*

Proof. Note that it is enough to show that the states $(n,0,0)$ and (x,y,zk) communicate. We prove that (x,y,zk) can be reached from $(n,0,0)$ in finite steps with non-zero probability by showing the following: we show that there indeed exists a path with the following intermediate states, each of, which can be reached in finite steps with non-zero probability:

$$\begin{aligned}&(n,0,0) \to (n-k,k,0) \to (n-y-zk, y+zk, 0)\\ &\to (n-y-zk, y+(l-1)k, k) \to (n-y-zk, y, zk)\ .\end{aligned}$$

We start by noticing that from $(n,0,0)$, $(n-k,k,0)$ is reached in k steps with probability 1. This is a direct consequence of the fact that there has to be at least k jobs at the batching station for batching to take place. Next, $(n-y-zk, y+zk, 0)$ can be reached in $y+(l-1)k$ steps where the probability of the r-th step is $p_r = \mathsf{P}(X_r < Y)$; X_r being an exponential variable with mean $1/((n-k-r+1)\lambda)$ and Y being another independent exponential variable with mean $1/M$. Each step here relates to the event that the client station sends a job to the batching station before a new batch is formed. Further, $(n-y-zk, y+(l-1)k, k)$ can be reached from $(n-y-zk, y+zk, 0)$ in a single step and the corresponding probability is $\mathsf{P}(Y < X)$. Here X is another independent exponential variable for which $\mathsf{E}[X] = 1/((n-y-zk)\lambda)$. This step corresponds to the event that a batch is formed by the batching station before it receives a new job. Finally, $(n-y-zk, y, zk)$ can be reached in another $(l-1)$ steps where the probability of the r-th step is $\mathsf{P}(Y < \min(X, Z_r))$; Z_r being another exponential variable with $\mathsf{E}[Z_r] = 1/(\min(n,r)\mu(k))$. Each intermediate step in this leg corresponds to the

event that the batching station prepares a batch before the client station could send a new job and the severs could serve a batch. A similar argument shows that starting from (x, y, zk), $(n, 0, 0)$ can be reached in finite steps with non-zero probability, which completes the proof. □

Note that the system described in Sect. 5.4.1. also attains a unique steady-state distribution and the proof of this fact follows similar arguments.

5.7.2 *System with Two Job Types and Non-preemptive Priority*

In contrast with the two job-type case with preemptive priority in Sect. 5.4, we additionally need the number of type 1 jobs in service for the case with non-preemption. The system state in this case can be represented as $(x, y_1, y_2, u_1k_1, v_1k_1)$ where x denotes the number of active clients, y_i denotes the number of type i jobs yet to be batched, u_1 denotes the number of type 1 batches in the queue and v_1 denotes the number of type 1 batches being served. Note that $s = (x, y_1, y_2, u_1k_1, v_1k_1)$ belongs to the state space:

$$\mathcal{S} = \left\{(x_1, x_2, x_3, x_4, x_5) \in \mathbb{Z}_+^5 : \mathbf{x.1} \leq n, k|x_4, k|x_5\right\} .$$

Here, the number of type 2 batches is: $z_2 = (n - s.\mathbf{1})/k_2$ out of which $v_2 = \min(m - v_1, z_2)$ are in service. The jump rates are of the CTMC are given by:

$$\begin{aligned}
s &\xrightarrow{\lambda xp} s - \mathbf{e_1} + \mathbf{e_2},\ x > 0 \\
&\xrightarrow{\lambda x(1-p)} s - \mathbf{e_1} + \mathbf{e_3},\ x > 0 \\
&\xrightarrow{M_1(k_1)\lfloor y_1/k_1 \rfloor} s - k_1\mathbf{e_2} + k_1\mathbf{e_4},\ y_1 \geq k_1, v = m \\
&\xrightarrow{M_1(k_1)\lfloor y_1/k_1 \rfloor} s - k_1\mathbf{e_2} + k_1\mathbf{e_5},\ y_1 \geq k_1, v < m \\
&\xrightarrow{M_2(k_2)\lfloor y_2/k_2 \rfloor} s - k_2\mathbf{e_2},\ y_2 \geq k_2 \\
&\xrightarrow{v_1\mu_1(k_1)} s + k_1\mathbf{e_1} - k_1\mathbf{e_5},\ v_1 \geq 1, u_1 = 0 \\
&\xrightarrow{v_1\mu_1(k_1)} s + k_1\mathbf{e_1} - k_1\mathbf{e_4},\ v_1 \geq 1, u_1 \geq 1 \\
&\xrightarrow{v_2\mu_2(k_2)} s + k_2\mathbf{e_1},\ v_2 \geq 1, u_1 = 0 \\
&\xrightarrow{v_2\mu_2(k_2)} s + k_2\mathbf{e_1} - k_1\mathbf{e_4} + k_1\mathbf{e_5},\ v_2 \geq 1, u_1 \geq 1,
\end{aligned} \tag{5.7.17}$$

where we have used the shorthand notation $s = (x, y_1, y_2, z_1k_1)$ and $v = v_1 + v_2$ is the total number of busy servers. The jump rates to all other states are zero. Similar to Sect. 5.4.1, we can get the steady state distribution $\boldsymbol{\pi}_0$ by solving

$\boldsymbol{\pi} \cdot \mathbf{Q} = 0$. Subsequently, we can derive the optimal throughput and find the corresponding optimal batch size.

5.7.3 *Multiple Job Types with Preemptive Priority*

Recall the setup described in Sect. 5.4.2 and assume that the total number of types of job is r. Further, job of type i_1 have preemptive priority over type i_2 in service whenever $i_1 < i_2$. With respect to producing jobs, we assume that each client generates a new a job of type i with probability p_i where $\sum p_i = 1$. Once a batch is formed, it goes through d levels of service and is sent back to the client station where it is un-batched and individual responses are channelled to respective clients. We further require that after each level of service, batches queue up in a common pool, specific to that particular level, until a server in the next stage[5] becomes available. Let us denote the batch size for job type i as k_i, which remains fixed across stages. Further, m_j denotes the total number of servers at stage j. Note that the service rate at stage j depends on the job type i and the corresponding batch size k_i and is denoted by $\mu_{ij}(k_i)$. We will omit the argument for μ_{ij} when the dependence is unambiguous.

Let us denote by $X_{ij}^{(n)}(t)$ the number of type i jobs that are at j-th level of service or are waiting for it. Clearly, $(X_{ij}^{(n)}(t), 1 \le i \le r, 1 \le j \le d, t \ge 0)$ is Markov on the state space $\mathcal{S} = \left\{(x_{ij}) \in \mathbb{Z}_+^{rd} : \sum_{i=1}^{r} \sum_{j=1}^{d} x_{i,j} \le n\right\}$. Observe that the quantity X_{1j} includes number of un-batched jobs of type j whereas X_{ij}, $i > 1$, is solely the number jobs that have formed batches. Let us now consider the corresponding scaled process $w^{(n)}(t) = (w_{ij}^{(n)}(t))$, $w_{ij}^{(n)}(t) = X_{ij}^{(n)}(t)/n$, $1 \le i \le r, 1 \le j \le d$. We see that

$$
\begin{aligned}
\dot{w}_{11} &= \lambda p_1 \left(1 - \sum_{a,b} w_{ab}\right) - \mu_{11} \min\left(k_1 \alpha_1, w_{11}\right) \ , \\
\dot{w}_{i1} &= \lambda p_i \left(1 - \sum_{a,b} w_{ab}\right) - \mu_{i1} \min\left(w_{i1}, \max\left(0, \alpha_1 - \sum_{l<i} \frac{w_{l1}}{k_l}\right) k_i\right) \ , \\
\dot{w}_{1j} &= \mu_{1(j-1)} w_{1(j-1)} - \mu_{1j} w_{1j} \ , \\
\dot{w}_{ij} &= \mu_{i(j-1)} \min\left(w_{i(j-1)}, \max\left(0, \alpha_{j-1} - \sum_{l<i} \frac{w_{l(j-1)}}{k_l}\right) k_i\right) \\
&\quad - \mu_{ij} \min\left(w_{ij}, \max\left(0, \alpha_j - \sum_{l<i} \frac{w_{lj}}{k_l}\right) k_i\right) \ , 2 \le i \le r \ , \quad 2 \le j \le d,
\end{aligned}
\tag{5.7.18}
$$

5 we use level/stage interchangeably

where $m_j/n \to \alpha_j$ and $\sum_{i,j} w_{ij} \leq 1$. Rewriting (5.7.18) as $\mathbf{F}(\mathbf{w}) = \frac{d\mathbf{w}}{dt}$, we note that $\mathbf{F}$ is Lipschitz continuous, which can be proved using similar arguments from part (i) of Thm. 1. In addition, the stationary measure $\pi_w^{(n)}$ is defined on the compact space $[0,1]^{rd}$, which implies tightness. We further see that the dynamical system in (5.7.18) is a piece-wise linear system and examine if it is globally attractive to the fixed point when all batch sizes are equal ($k_i = k$) and the system has only one level of service ($d = 1$).

Theorem 3. If $k_i = k$, $1 \leq i \leq r$ and $d = 1$, the system in (5.7.18) is globally attractive to

$$\mathbf{w}^* = \begin{cases} \mathbf{A}^{-1}\mathbf{c_a}\,, & \text{if } \langle \mathbf{A}^{-1}\mathbf{c_a}, 1\rangle < k\alpha \\ \mathbf{B}^{-1}\mathbf{c_b}\,, & \text{otherwise,} \end{cases}$$

where

$$\mathbf{A} = \begin{bmatrix} -\mu_1 - \lambda p_1 & -\lambda p_1 & \dots & -\lambda p_1 \\ -\mu_2 & -\mu_2 - \lambda p_2 & \dots & -\lambda p_2 \\ \dots & \dots & \dots & \dots \\ -\mu_r & -\mu_r & \dots & -\mu_r - \lambda p_r \end{bmatrix},\ \mathbf{c_a} = \begin{bmatrix} -\lambda p_1 \\ -\lambda p_2 \\ \dots \\ -\lambda p_r \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} -\mu_1 - \lambda p_1 & -\lambda p_1 & \dots & -\lambda p_1 \\ -\mu_2 & -\mu_2 - \lambda p_2 & \dots & -\lambda p_2 \\ \dots & \dots & \dots & \dots \\ \mu_r - \lambda p_r & \mu_r - \lambda p_r & \dots & -\lambda p_r \end{bmatrix},\ \mathbf{c_b} = \begin{bmatrix} -\lambda p_1 \\ -\lambda p_2 \\ \dots \\ -\lambda p_r + k\alpha\mu_r \end{bmatrix}.$$

Proof. For $d = 1$, the service stage index j can be omitted. Now, using $k_i = k$ and $\alpha_1 = \alpha$, the ODEs from (5.7.18) can be written as:

$$\begin{aligned} \dot{w}_1 &= \lambda p_1 \left(1 - \sum_a w_a\right) - \mu_1 \min\left(k\alpha, w_1\right), \\ \dot{w}_i &= \lambda p_i \left(1 - \sum_a w_a\right) - \mu_i \min\left(w_i, \max\left(0, k\alpha - \sum_{l<i} w_l\right)\right), 2 \leq i \leq r. \end{aligned} \tag{5.7.19}$$

We now prove that $\mathbf{B}$ is non-singular and real parts of the eigenvalues of $\mathbf{B}$ are negative. Similar arguments hold for $\mathbf{A}$; although the calculations turn out to be simpler. Let $\mathbf{Bx} = \mathbf{0}$ with $\mathbf{x} \neq \mathbf{0}$, i.e.,

$$\begin{bmatrix} \mu_1 x_1 \\ \mu_2 x_2 \\ \dots \\ \mu_r x_r \end{bmatrix} = -\sum_l x_l \begin{bmatrix} \lambda p_1 \\ \lambda p_2 \\ \dots \\ \lambda p_r - \mu_r \end{bmatrix}.$$

Since $\mathbf{x} \neq \mathbf{0}$, $\sum_l x_l \neq 0$ and

$$\sum_l x_l = -\lambda \left(\sum_l x_l \right) \left(\sum_l \frac{p_l}{\mu_l} \right) + \sum_l x_l$$
$$\text{i.e.,} \quad \lambda \sum_l \frac{p_l}{\mu_l} = 0.$$

This contradicts our assumption that λ, μ_i's and p_i's are positive.

Next we show that all eigenvalues of $\mathbf{B}$ have negative real part. Let θ be an eigenvalue of $\mathbf{B}$ and $\mathbf{u}$ be one of its eigenvectors. If $\theta = -\mu_j$ for some j, our hypothesis is trivially true. Therefore, we can focus on the scenario where $\theta \neq -\mu_j \ \forall j$. This implies

$$u_j(\theta + \mu_j) = -\lambda p_j \sum_l u_l, \quad 1 \leq j \leq r-1$$
$$\text{and} \quad u_r(\theta + \mu_r) = (-\lambda p_r + \mu_r) \sum_l u_l.$$

Since $\theta \neq -\mu_j \ \forall j$ and $\mathbf{u} \neq \mathbf{0}$, we see that $\sum_l u_l \neq 0$ and

$$\sum_l u_l = \left(-\frac{\mu_r}{\mu_r + \theta} + \sum_l \frac{-\lambda p_l}{\mu_l + \theta} \right) \left(\sum_l u_l \right),$$
$$\text{i.e.,} \ \sum_l \frac{\lambda p_l}{\mu_l + \theta} = -1 + \frac{\mu_r}{\mu_r + \theta},$$
$$\text{i.e.,} \ \sum_l \frac{\lambda p_l (\mu_l + \Re(\theta))}{|\mu_l + \theta|^2} = -1 + \frac{\mu_r (\mu_r + \Re(\theta))}{|\mu_r + \theta|^2}.$$

Unless $\Re(\theta) < 0$, the left and right hand sides have different signs. Thus we have shown that $\mathbf{B}$ (and by similar arguments, $\mathbf{A}$) is non-singular and real parts of its eigenvalues are negative. We use this fact to show that the system in (5.7.19) is globally attractive to the unique fixed point. We consider the following scenarios.

Case 1: $k\alpha \geq 1$

In this case, (5.7.19) can be written as

$$\frac{d\mathbf{w}}{dt} = \mathbf{A}\mathbf{w} - \mathbf{c_a}.$$

Since $\mathbf{A}$ is non-singular and all eigenvalues of $\mathbf{A}$ have negative real part, the system is globally attractive to the unique fixed point $\mathbf{A}^{-1}\mathbf{c_a}$.

Case 2: $k\alpha < 1$

We first show that $\langle \mathbf{A}^{-1}\mathbf{c_a}, 1 \rangle < k\alpha$ iff $\langle \mathbf{B}^{-1}\mathbf{c_b}, 1 \rangle < k\alpha$. For if $\mathbf{A}\mathbf{x} = \mathbf{c_a}$ and $\mathbf{B}\mathbf{y} = \mathbf{c_b}$, we see that

$$\sum_l x_l = \frac{\sum_l \frac{\lambda p_l}{\mu_l}}{1 + \sum_l \frac{\lambda p_l}{\mu_l}} \text{ and } \sum_l y_l = \frac{\sum_l \frac{\lambda p_l}{\mu_l} - k\alpha}{\sum_l \frac{\lambda p_l}{\mu_l}}.$$

Further,

$$\frac{\sum_l \frac{\lambda p_l}{\mu_l}}{1+\sum_l \frac{\lambda p_l}{\mu_l}} < k\alpha \iff \frac{\sum_l \frac{\lambda p_l}{\mu_l} - k\alpha}{\sum_l \frac{\lambda p_l}{\mu_l}} < k\alpha.$$

Now, we notice that the state of the system eventually belongs to the region $\sum_{i\leq r-1} w_i < k\alpha$. For if there exists a lowest index $i_0 < r-1$ with $\sum_{i\leq i_0} w_i \geq k\alpha$, we have $\frac{dw_i}{dt} \geq 0$ for $i \geq i_0$, as the domain of interest is $\sum_i w_{ij} \leq 1$. This entails instability because w_i, $i > i_0$ increases forever. Hence the system eventually enters the region $\sum_{i\leq r-1} w_i < k\alpha$.

Let us first consider the scenario where $\langle \mathbf{A}^{-1}\mathbf{c_a}, 1\rangle < k\alpha$. If the initial state of the system falls in the subregion $\sum_{i\leq r} w_i < k\alpha$, the system evolves in a way analogous to the case $k\alpha \geq 1$ and thus converges to $\mathbf{A}^{-1}\mathbf{c_a}$. On the contrary, starting the system from subregion $\sum_{i\leq r} w_i \geq k\alpha$ causes the system to evolve according to

$$\frac{d\mathbf{w}}{dt} = \mathbf{B}\mathbf{w} - \mathbf{c_b}.$$

This implies that the system drifts towards $\mathbf{B}^{-1}\mathbf{c_b}$ as the eigenvalues have negative real part ($\mathbf{B}^{-1}$ exists as $\mathbf{B}$ is non singular). We have already shown that $\langle \mathbf{A}^{-1}\mathbf{c_a}, 1\rangle < k\alpha$ implies $\langle \mathbf{B}^{-1}\mathbf{c_b}, 1\rangle < k\alpha$. Therefore, the system eventually enters the subregion $\sum_{i\leq r} w_i < k\alpha$, which guarantees convergence to $\mathbf{A}^{-1}\mathbf{c_a}$. For the scenario $\langle \mathbf{A}^{-1}\mathbf{c_a}, 1\rangle > k\alpha$, we can similarly show that the system converges to $\mathbf{B}^{-1}\mathbf{c_b}$. □

6 CONSTRAINED THROUGHPUT OPTIMIZATION

6.1 INTRODUCTION

In Chap. 3–5, we considered the impact of transitions on the performance of different TBSs connected to a typical present-day network to answer the first two research questions posed in this thesis. In this chapter, we move on to our third research question which revolves around the overall networked system quality. We first note that the functioning of systems connected to a network is supported by basic network functionalities such as packet forwarding, routing and access control. To that end, we consider a network switch, which is a fundamental network element connecting multiple systems and performing some of the aforementioned functionalities. Most modern network switches have built-in buffers, which queue data packets while the switch is busy processing and transmitting packets to the output ports. A large buffer has the advantage that it can potentially minimize packet loss, i.e., dropping of packets waiting to be processed when the buffer can no longer hold them. However, in a busy network, large buffers often lead to excess buffering of packets, a phenomenon that is commonly referred to as bufferbloat (Gettys 2011). While a shallow buffer alleviates this phenomenon, it can also lead to frequent packet drops, which cause TCP flows[1] to die (Chiu and Jain 1989). To get around this problem, switches actively decide whether to drop an incoming packet depending on the current buffer-filling and recent packet drop or packet delay history– referred to as Active Queue Management (AQM). Put another way, AQM facilitates transitions in the state of the switch, which helps it achieve a desired level of performance. In the last three decades, a range of AQM algorithms have been proposed (Adams 2013; Floyd and Jacobson 1993; Nichols and Jacobson 2012; Pan, Natarajan, et al. 2013b), which are either heuristic-based or require experimental parameter engineering (Khademi, Ros, et al. 2014).

In this chapter, we take a direct approach to solve the AQM problem. We formulate a framework for the state evolution of the switch together with a precise optimization goal to derive a policy for packet drop. Further, compared to the state of the art, which look at the history of packet delay or packet drop on top of the current backlog at the switch, we formulate our policy based on the current rate of packet arrivals. Our choice is motivated by the fact that the arrival rate has

1 We consider only TCP flows as it constitutes more than 80% of the internet traffic according to some estimates (Lee, Carpenter, et al. 2010).

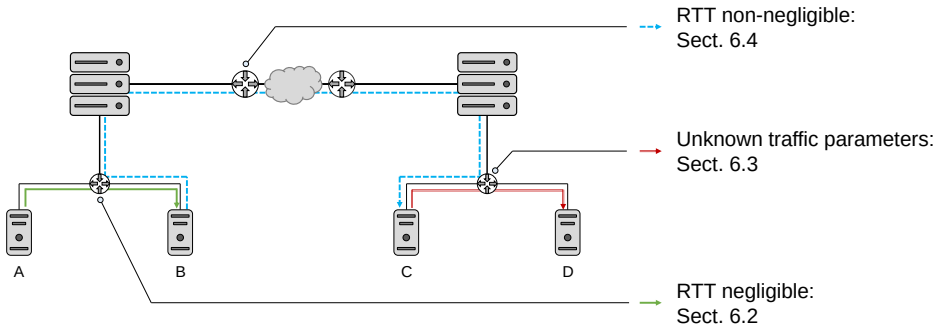


Figure 6.1: A representative example of two datacenters connected over the internet: while a TCP flow between two nodes of the same hierarchy in a datacenter (e.g., A and B) has negligible RTT, an inter-datacenter flow (e.g., between A and C) is likely to have an RTT which is significant. We derive the policy for a switch catering to a flow with negligible RTT and known packet arrival characteristics in Sect. 6.2. When traffic properties are unknown, we estimate corresponding parameters as per Sect. 6.3. For flows with non-negligible RTT, we revise our approach as outlined in Sect. 6.4.

an immediate consequence on the congestion at the buffer. To that end, we first model the state evolution of a switch. We encode the optimization goal through a reward function that reflects the immediate gain following a drop/admit action. Subsequently, we derive an optimal policy using tools from Markov Decision Processes (MDP) that use the state transition probabilities calculated from the model and the reward function as inputs. Finally, we compare the performance of our policy against CoDel, the most prevalent AQM algorithm and the traditional approach of drop tail queues.

6.2 AQM AS AN OPTIMAL DECISION PROBLEM

In this section, we formulate the task of AQM in real systems as deriving an optimal policy of a Semi Markov Decision Process. We first observe that the problem can be formulated in different ways based on the flow RTT and flow properties. Such inherent distinctions, e.g., with respect to flow RTT can arise during intra-datacenter or inter-datacenter communications as shown in Fig. 6.1. In case of negligible RTT, our model allows for a more general arrival process whereas a simpler model is adopted for the case with non-negligible RTT for better tractability. We discuss the former in greater detail here and take up the latter in Sect. 6.4.

Under the scenario of negligible RTT, the buffer is observed at arrival instants[2] and we seek to derive the optimal policy, i.e., whether it is ideal to drop or admit a packet given the current state of the system. Other than action, the evolution of the system is determined by the packet interarrival time and the service time distributions. We assume that interarrival times are gamma distributed with parameters dependent upon the history. Gamma interarrival times allows us to model a wide class of empirical traffic flows. Further, the service times are assumed to be exponentially distributed. Before going into the details of the system, we emphasize that we define optimality in terms of expected average long term reward. Expected average long term reward can be viewed as long term accumulation of instant rewards per unit time where the instant reward can be fed in to convey the ultimate objective such as throughput and/or delay. In the following, we provide the notations required to describe the AQM problem:

Notations:

Q_t: queue length immediately before the arrival of the t-th packet,
α: common shape parameter of the Gamma variable describing the packet interarrival time distributions,
β_t: gamma rate parameter of the packet interarrival time during the arrival of the t-th packet,
μ: service rate of the packets,
S_t: state of the system observed by the packet corresponding to t-th arrival, defined as (Q_t, β_t),
A_t: action taken upon t-th arrival,
$\mathsf{P}\left(S_{t+1}|S_t, A_t\right)$: transition probability to state S_{t+1} from S_t given action A_t was taken,
$R(S_t, A_t)$: expected (immediate) reward when action A_t is taken in state S_t,
$\tau(S_t, A_t)$: expected transition time when action A_t is taken in state S_t,
X_t: time of t-th packet arrival.

Let us denote the state space by $\mathcal{S}$ and the action space by $\mathcal{A} = \{0, 1\}$. In $\mathcal{A}$, the action 0 denotes admitting a packet and 1 denotes a drop. We use the notations here across sections with subtle variations, which is specified in respective contexts. For example, in Sect. 6.4 where the interarrival times are assumed to be exponentially distributed and the RTT is considered non-negligible, β_t denotes the rate parameter of the interarrival times *at* t-th arrival.

Looking at the consequence of possible actions, we see that letting a packet in entails an immediate increase in the queue length by one and the effective arrival rate β_t/α of the TCP flow increases to $f_u(\beta_t/\alpha)$ instantaneously[3]. Here, the

2 We use instant and epoch interchangeably.

3 On the contrary, the change in arrival rate occurs with a delay when RTT is non-negligible; see Sect. 6.4.

function f_u is dependent upon the underlying TCP congestion control algorithm. For example, for an additive increase multiplicative decrease (AIMD) algorithm (Chiu and Jain 1989), $f_u(\beta_t/\alpha) = \beta_t/\alpha + a$, for some $a > 0$. For the ease of formulation, we assume that the shape parameter α remains fixed and modify the rate parameter β appropriately to reflect this change. Thus following an admit action, β_{t+1} assumes the value $\alpha f_u(\beta_t/\alpha)$.

In contrast, packet drops clearly do not change the queue length although the effective arrival rate β_t/α decreases to $f_d(\beta_t/\alpha)$ immediately where f_d depends upon the underlying TCP congestion control algorithm. For an AIMD implementation, $f_d(\beta_t/\alpha) = b\beta_t/\alpha$, for some $0 < b < 1$. As before, we keep the shape parameter fixed and take $\beta_{t+1} = b\beta_t$. For simplicity of notation, we derive results for the simplest version of AIMD algorithm, i.e., $a = 1$ and $b = 1/2$. Our results can be generalized by replacing the increment and decrement of flow arrival rate with corresponding f_u and f_d of the underlying congestion control algorithm.

Observe that under the simple AIMD algorithm, the action $A_t = 0$ leads to an immediate jump in the queue length to $(Q_t + 1)$ and the possible states in the next arrival instant could be any member of the set: $\mathcal{Q}_0 = \{(q, \beta_t + \alpha) : 0 \leq q \leq Q_t + 1\}$. That is, $S_{t+1} \in \mathcal{Q}_0$ and $\mathsf{P}\left((q, \beta_t + \alpha)|S_t, 0\right)$ denotes the probability that exactly $Q_t + 1 - q$ packets are served until the next packet arrival. Similarly, for the action $A_t = 1$, we see that $\mathcal{Q}_1 = \{(q, \beta_t/2) : 0 \leq q \leq Q_t\}$ and $\mathsf{P}\left((q, \beta_t/2)|S_t, 1\right)$ is the probability of serving $Q_t - q$ packets until the next arrival. The transition probabilities to any state outside $\mathcal{Q}_0$ and $\mathcal{Q}_1$ are zero. Next, we derive the expression for state transition probabilities, which are crucial inputs to the optimal policy derivation process.

6.2.1 *State Transition Probabilities*

Recall that we described the system state as the tuple of the present queue length and the arrival rate, i.e., $S_t = (Q_t, \beta_t)$. Here we derive the expression for the transition probabilities $\mathsf{P}\left(S_{t+1}|S_t, A_t\right)$, which are inputs to the Bellmann operator. This operator iteratively determines the *value* of each state, which in turn determines the policy; see Sect. 7.1 of (Tijms 2003) for details. The following lemma is a first step towards the derivation.

Lemma 4. For two independent Gamma distributed variables, $Y_{u,v} \sim \text{Gamma}(u, v)$ and $X_{w,z} \sim \text{Gamma}(w, z)$,

$$\mathsf{P}\left(Y_{u,v} > X_{w,z}\right) = \mathsf{P}\left(Y_{u-1,v} > X_{w,z}\right) + \frac{\Gamma(u+w-1)}{\Gamma(u)\Gamma(w)}\left(\frac{v}{v+z}\right)^{u-1}\left(\frac{z}{v+z}\right)^{w},$$

where $u > 1$. This further implies

$$\mathsf{P}\left(Y_{u,v} > X_{w,z}\right) = \sum_{k=0}^{u-1} \frac{\Gamma(k+w)}{\Gamma(k+1)\Gamma(w)} \left(\frac{v}{v+z}\right)^k \left(\frac{z}{v+z}\right)^w.$$

Proof. See Sect. 6.7. □

Using the lemma, we derive the transition probabilities as follows.

Theorem 5. Let $S_t = (Q_t, \beta_t)$ denote the system state at t-th arrival instant, Q_t and β_t being the current queue length and arrival rate, respectively. Under action $A_t = 0$, the transition probabilities to the state $S_{t+1} = (Q_{t+1}, \beta_{t+1})$ in the next arrival instant are given by

$$\mathsf{P}\left((q, \beta_{t+1})|S_t, 0\right) = \mathbb{1}_{\beta_{t+1}=\beta_t+\alpha} \frac{\Gamma(Q_t+1-q+\alpha)}{\Gamma(Q_t+2-q)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha},$$

for $1 \leq q \leq Q_t + 1$, and

$$\mathsf{P}\left((0, \beta_{t+1})|S_t, 0\right) = \mathbb{1}_{\beta_{t+1}=\frac{\beta_t}{2}} \left(1 - \sum_{k=0}^{Q_t} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha}\right).$$

Here we have used the shorthand notation $q = Q_{t+1}$.

Proof. Observe that $P((q, \beta_{t+1})|S_t, 0)$ is the probability of the event that following an admit action, exactly $Q_t + 1 - q$ packets are served between two packet arrivals. We further see that serving at least n packets between two arrivals is equivalent to the event that the corresponding service time $Y_{n,\mu}$ is less than the interarrival time $X_{\alpha,\beta_{t+1}}$. The value of arrival rate parameter β_{t+1} is explained by the fact that following an admit action on t-th packet arrival, the rate parameter immediately changes to $\beta_{t+1} = \beta_t + \alpha$, which dictates the distribution of the next interarrival time. Hence, given $\beta_{t+1} = \beta_t + \alpha$ and $1 \leq q \leq Q_t + 1$,

$$\begin{aligned}
\mathsf{P}\left((q, \beta_{t+1})|S_t, 0\right) &= \mathsf{P}\left(Y_{Q_t+2-q,\mu} > X_{\alpha,\beta_{t+1}}\right) - \mathsf{P}\left(Y_{Q_t+1-q,\mu} > X_{\alpha,\beta_{t+1}}\right) \\
&= \frac{\Gamma(Q_t+1-q+\alpha)}{\Gamma(Q_t+2-q)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha}.
\end{aligned}$$

For $q = 0$, we see that at most $Q_t + 1$ packets can be served between two arrivals. Hence, given $\beta_{t+1} = \beta_t + \alpha$,

$$\begin{aligned}
\mathsf{P}\left((0, \beta_{t+1})|S_t, 0\right) &= \mathsf{P}\left(Y_{Q_t+1,\mu} \leq X_{\alpha,\beta_{t+1}}\right) \\
&= 1 - \mathsf{P}\left(Y_{Q_t+1,\mu} > X_{\alpha,\beta_{t+1}}\right) \\
&= 1 - \sum_{k=0}^{Q_t} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha}.
\end{aligned}$$

□

We take a similar approach to derive the transition probabilities for drop action. In this case, only the rate parameter of the interarrival times changes immediately to $\beta_t/2$ and the possible queue length in the next decision epoch is no more than Q_t. Hence,

$$\mathsf{P}\left((q,\beta_{t+1})|S_t,1\right) = \mathbb{1}_{\beta_{t+1}=\frac{\beta_t}{2}} \frac{\Gamma(Q_t-q+\alpha)}{\Gamma(Q_t+1-q)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha},$$

for $1 \leq q \leq Q_t$, and

$$\mathsf{P}\left((0,\beta_{t+1})|S_t,1\right) = \mathbb{1}_{\beta_{t+1}=\frac{\beta_t}{2}} \left(1 - \sum_{k=0}^{Q_t-1} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^{\alpha}\right).$$

In contrast to the Markov Decision Process framework, the expected time between two decision epochs is crucial for deriving the optimal policy in the SMDP framework. Since the drop or admit action influences the packet sending rate of TCP, the expected time between two arrival epochs varies. We note that the expected transition times are: $\tau(S_t,0) = \alpha/(\beta_t+\alpha)$ and $\tau(S_t,1) = 2\alpha/\beta_t$, following the AIMD principle of TCP. We abbreviate $\tau(S_t,A_t)$ as τ_t when the context is clear. We illustrate the state transitions through a diagram in Fig. 6.2.

6.2.2 *Reward Function*

Equipped with the transition probabilities, we now focus on the reward function, which conveys the objective to the policy learning process. The reward function describes the immediate incentive and the learning process aims to maximize the mean cumulative reward in the long run. Since we want our AQM policy to maximize throughput given a constraint on delay, we choose a reward function that imposes a heavy penalty at the breach of the delay constraint and grants an immediate incentive amounting to the change in the square rate of throughput. The rationale here is to represent a measure of relative rather than absolute change. Formally,

$$\begin{aligned} R(S_t,0) &= -M\mathbb{1}_{\{(Q_t+1)/\mu>\eta\}} + \left(\sqrt{\frac{\beta_t+\alpha}{\alpha}} - \sqrt{\frac{\beta_t}{\alpha}}\right), \\ R(S_t,1) &= -M\mathbb{1}_{\{Q_t/\mu>\eta\}} + \left(\sqrt{\frac{\beta_t}{2\alpha}} - \sqrt{\frac{\beta_t}{\alpha}}\right), \end{aligned} \tag{6.2.1}$$

where η denotes a user-defined target delay and M represents the penalty amount for breaching it, which is typically large. Note that the reward function mimics state-of-the-art heuristic AQM algorithms where the target delay is the single input variable (Nichols, Jacobson, et al. n.d.).

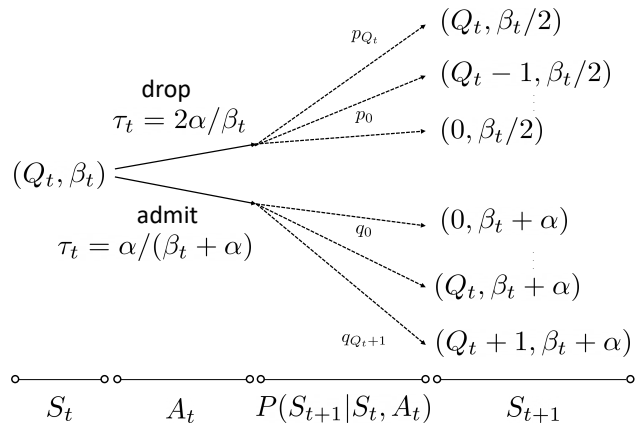


Figure 6.2: State transition from state (Q_t, β_t) (Q_t: current buffer filling, β_t: current arrival rate parameter) after packet admission or drop. Solid arrows represent actions whereas dotted arrows show possible state transitions annotated with corresponding probabilities. τ_t denotes the expected time until the next decision epoch, i.e., the mean interarrival time.

The transition probabilities and the reward function are the primary inputs to the value-iteration algorithm, which computes the optimal policy. This policy runs on the buffer and takes the system state (Q_t, β_t), i.e., the current backlog and the arrival rate parameter as input and returns the optimal action (drop/admit). Recall that optimality was defined in terms of average cumulative reward in the long run.

Formally, a policy π is defined as a mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$, i.e., given the system state in terms of buffer filling and arrival rate, the policy returns one of the possible actions- admit or drop. Let us denote the cumulative reward up to time x by $Z(x)$, i.e.,

$$Z(x) = \sum_{t=1}^{T(x)} R(S_t, A_t) \ , \quad \text{where}$$
$$T(x) = \max \{t : X_t \leq x\} \ ,$$

and X_t denotes the arrival time of the t-th packet. Then, the expected average long-term reward is defined as:

$$g_i(\pi) = \lim_{x \to \infty} \frac{1}{x} \mathsf{E}_{i,\pi}[Z(x)] \ , \tag{6.2.2}$$

where i represents the initial state S_i and π signifies the used policy.

Recall that our objective was to derive a policy π that maximizes g for each state $s \in \mathcal{S}$. To that end, we translate the SMDP into a discrete-time MDP using

the data transformation method (Tijms 2003). This method scales the rewards appropriately and transforms the transition probabilities. It relies on the fact that an SMDP with average reward criterion is equivalent to a discrete time MDP with rewards being the reward accumulation rate of the SMDP and with appropriately adjusted transition probabilities. Subsequently, the value iteration method for discrete time MDP's can be used to compute the optimal policy. The transformed reward function $\bar{R}$ and the transition probability $\bar{\mathsf{P}}$ for the discrete-time MDP are given by:

$$\begin{aligned}
\bar{R}(S_t, A_t) &= \frac{R(S_t, A_t)}{\tau(S_t, A_t)}\,, \\
\bar{\mathsf{P}}(S_{t+1}|S_t, A_t) &= \mathsf{P}(S_{t+1}|S_t, A_t)\frac{\tau}{\tau(S_t, A_t)},\ S_{t+1} \neq S_t\,, \\
\bar{\mathsf{P}}(S_t|S_t, A_t) &= \mathsf{P}(S_t|S_t, A_t)\frac{\tau}{\tau(S_t, A_t)} + 1 - \frac{\tau}{\tau(S_t, A_t)}\,,
\end{aligned} \tag{6.2.3}$$

where τ is chosen subject to the following condition: $0 < \tau \leq \min_{s,a} \tau(s, a)$.

6.3 AQM FOR UNKNOWN TRAFFIC FLOW PROPERTIES

In this section, we consider the scenario where traffic flow properties are unknown. As seen in the Sect. 6.2, the optimal policy is dependent upon the current traffic arrival rate, which is determined by the initial arrival rate and the consequent series of actions (drop/admit). Using the information on the current arrival rate and the queue filling, the AQM policy prescribes whether it is optimal to drop or admit a packet. Therefore, to propose a policy under unknown arrival characteristics, we first need to infer the respective parameters. However, estimating the model parameters while learning the optimal policy introduces the problem of dual control (Feldbaum 1960). This can be alleviated by an exploration-exploitation Bandit-heuristic (Sutton and Barto 2018), which is computationally tractable.

6.3.1 *Estimation of Arrival Parameters*

In this section, we describe the method to infer the parameters governing the arrival process when the congestion control algorithm is known. Recall from Sect. 6.2 that the interarrival times are Gamma distributed and we calculate Maximum Likelihood Estimates (MLEs) of the arrival shape parameter α and rate parameter β_t. Let us denote by $\{X_n\}$ and $\{A_n\}$ the sequence of arrival times and the actions, respectively. Corresponding packet interarrival times are denoted by

W_t, i.e., $W_t = X_{t+1} - X_t$. Recall that $A_t = 1$ implies a packet drop and $A_t = 0$ corresponds to admitting a packet. We assume

$$\begin{aligned} W_t|\alpha, \beta_t &\sim \text{Gamma}(\alpha, \beta_t) \,, \\ \beta_{t+1} &= (\beta_t + \alpha)^{1-A_t} (\frac{\beta_t}{2})^{A_t} \,, \end{aligned} \tag{6.3.4}$$

following the AIMD congestion control principle of TCP. For the sequence of interarrival times $\mathbf{W} = \{W_n\}_{n=1}^k$ and actions $\mathbf{A} = \{A_n\}_{n=1}^k$, we seek to estimate the common shape parameter α and the initial rate parameter β_1. This lets us calculate the transition probabilities $P(S_{t+1}|S_t, A_t)$ described in Sect. 6.2, which are inputs to the optimal policy derivation process. The likelihood of the parameters given the observed sequence $(\mathbf{W}, \mathbf{A})$ is:

$$L(\alpha, \beta_1|\mathbf{W}, \mathbf{A}) = \frac{(\prod_{n=1}^k \beta_n)^\alpha (\prod_{n=1}^k W_n))^{\alpha-1} e^{-\sum_{n=1}^k \beta_n W_n}}{(\Gamma\alpha)^k} \,.$$

Therefore, the log-likelihood $l(\alpha, \beta_1)$ is given by:

$$l(\alpha, \beta_1) = \alpha \sum \log \beta_n + (\alpha - 1) \sum \log W_n - \sum \beta_n W_n - k \log(\Gamma\alpha) \,. \tag{6.3.5}$$

This implies

$$\begin{aligned} \frac{\partial l}{\partial \alpha} &= \sum \log \beta_n + \sum \left(\frac{\alpha}{\beta_n} - W_n \right) \frac{\partial \beta_n}{\partial \alpha} + \sum \log W_n - k \frac{\partial}{\partial \alpha} \log(\Gamma\alpha) \,, \\ \frac{\partial l}{\partial \beta_1} &= \sum \left(\frac{\alpha}{\beta_n} - W_n \right) \frac{\partial \beta_n}{\partial \beta_1} \,, \end{aligned} \tag{6.3.6}$$

where

$$\frac{\partial \beta_n}{\partial \alpha} = \left(1 + \frac{\partial \beta_{n-1}}{\partial \alpha} \right)^{1-A_{n-1}} \left(\frac{1}{2} \frac{\partial \beta_{n-1}}{\partial \alpha} \right)^{A_{n-1}} \,, \qquad n = 3, 4, \ldots, k,$$

$\partial\beta_2/\partial\alpha = \mathbb{1}_{\{A_1=0\}}$ and $0^0 = 1$ by convention. Further,

$$\frac{\partial \beta_n}{\partial \alpha} = \left(\frac{\partial \beta_{n-1}}{\partial \beta_1} \right)^{1-A_{n-1}} \left(\frac{1}{2} \frac{\partial \beta_{n-1}}{\partial \beta_1} \right)^{A_{n-1}} = \left(\frac{1}{2} \right)^{A_{n-1}} \frac{\partial \beta_{n-1}}{\partial \beta_1} \,, \quad n = 2, 3, 4, \ldots, k \,.$$

To find the MLEs $\hat{\alpha}$ and $\hat{\beta_1}$, we numerically find the zeros of (6.3.6) that maximizes (6.3.5).

6.3.2 *Inference under unknown TCP Congestion Control*

In case the underlying congestion control algorithm is not known to the user, we assume that the effective arrival rate β_{t+1}/α in the next arrival instance evolves

according to a polynomial function (e.g., CUBIC (Ha, Rhee, et al. 2008)) of the current effective arrival rate β_t/α. Thus, (6.3.4) can be rewritten as

$$\begin{aligned} W_t|\alpha,\beta_t &\sim \text{Gamma}(\alpha,\beta_t),\\ \frac{\beta_{t+1}}{\alpha} &= \left(f_u\left(\frac{\beta_t}{\alpha}\right)\right)^{1-A_t}\left(f_d\left(\frac{\beta_t}{\alpha}\right)\right)^{A_t},\\ \text{i.e.,}\quad \beta_{t+1} &= \alpha\left(f_u\left(\frac{\beta_t}{\alpha}\right)\right)^{1-A_t}\left(f_d\left(\frac{\beta_t}{\alpha}\right)\right)^{A_t}. \end{aligned} \tag{6.3.7}$$

Here f_u and f_d are polynomials that represent the change in effective arrival rate following an admit and drop, respectively. The log-likelihood, in this case, has a similar form, although we require the estimates of the coefficients of the polynomials along with estimates of α and β_1. The estimates can be computed numerically or by calculating the partial derivatives explicitly, similar to (6.3.6). We can then compute their zeros corresponding to maxima, although with additional equations for $\partial l/\partial c_j$, for each coefficient of the polynomials c_j. For example, if f_u is a cubic polynomial and f_d is a linear function, the calculation of six additional partial derivatives is required.

Using the parameter estimates along with the observed sequence of actions (drop/admit), we infer the current arrival rate $\hat{\beta}_t$. Together with the present queue filling Q_t, it describes the present state of the system. The AQM policy can now be used to find the corresponding optimal action.

6.4 AQM UNDER NON-NEGLIGIBLE RTT

In this section, we consider the case where the RTT of a flow is non-negligible. Recall from Fig. 6.1 that a corresponding example would be a flow between a sender and a receiver belonging to different datacenters. As mentioned in Sect. 6.2, we adopt a simpler model in this case for tractability. This is particularly applicable to the scenario where the switch has to act on multiple incident flows. We note that for a single flow, the analysis is relatively simpler, which we take up in Sect. 6.4.1. The multiple flow case is described in Sect. 6.4.2. Similar to the previous case, our objective is to maximize total throughput adhering to the given delay constraint. Throughout, we assume that the flow RTTs are known to the switch.

6.4.1 *AQM under Single Flow*

In the following, we assume that the flow interarrival times and the service times are exponentially distributed. We denote the flow RTT as $r \in \mathbb{R}_+$. Unlike the previous framework, the sender modifies the sending rate (arrival rate to the switch) no earlier than it receives a congestion signal from the receiver. The

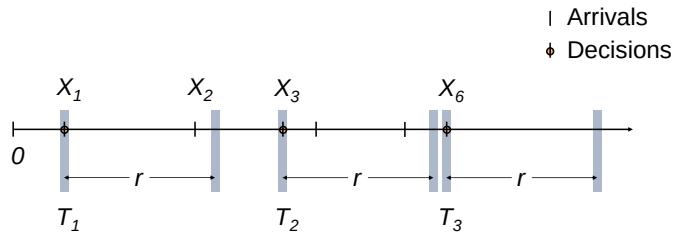


Figure 6.3: Arrival and decision times when the switch deals with a single flow: the first job arrives at X_1 with rate β_1. X_2 is the only arrival in $(X_1, X_1 + r]$ and has no effect on the arrival rate as $X_2 < X_1 + r$. Note that $T_2 = X_3$ and the arrival rate changes to β_2 at T_2 according to action A_1 taken at X_1. The next change in the rate occurs at $T_3 = X_6$ where it updates to β_3.

signal takes $r/2$ time to reach the sender. However, the delay between an action at the switch and the arrival of a congestion signal includes the propagation time between the switch and the receiver on top of this $r/2$ amount of time. We approximate the corresponding total time by the RTT r. This has the following implication for the framework: an action takes effect only after a time period r, and all arrivals in this period are admitted subject to sufficient room in the buffer. For the ease of formulation, we additionally assume that these intermediate arrivals (within the time period r) and corresponding admit actions do not influence the arrival rate. We do not expect this *approximation* to have a major impact on the analysis as an admit action causes the arrival rate to jump by only a packet per second. However, this assumption considerably simplifies the representation of the state of the switch as it lets us discard the admit actions taken on these arrivals.

As before, we denote the arrival times as $\{X_s\}$, $s \in \mathbb{N}$. The first decision is taken at the first packet arrival at time X_1. All subsequent arrivals in $(X_1, X_1 + r]$ are admitted given there is sufficient room in the buffer. The next drop decision can be potentially taken on the first arrival after $(X_1 + r)$. At the first arrival after $(X_1 + r)$, the arrival rate changes according to the action taken at time X_1. Subsequently, the decision-making process stays dormant for another r time period and the next drop decision can be possibly taken again at the first arrival after this period. Needless to say, this packet can also be admitted if the decision rule prescribes so. Further, we see a change in the arrival rate at this time according to the action taken in the previous decision epoch. Note that a decision epoch or a decision time is an arrival epoch where a drop decision can be possibly made. We denote decision epochs as $\{T_t\}$, $t \in \mathbb{N}$. To state the obvious, the arrival rate changes according to the action taken at T_{i-1} at the time point T_i ($i \geq 2$). Note that given the relevant arrival rate parameter β_k,

$$X_{i+1} - X_i \sim \text{Exp}(\beta_k),\ i \in \mathbb{N},$$

for some $k \in \mathbb{N}$. Further, unlike Sect. 6.2, the arrival rates do not change at every arrival. Instead, we see a change in the arrival rate only at the decision times due to the delay factor of r and the fact that the arrivals between two decision times do not influence the arrival rate directly. For the inter-decision time, we see that the time until next decision time after one RTT r has passed, is distributed as $\text{Exp}(\beta_j)$, for some $j \in \mathbb{N}$, due to the memoryless property of exponential distribution. Using the fact that the arrival rate β changes only at the decision times, we can write

$$T_{i+1} - T_i | \beta_i \sim r + \text{Exp}(\beta_i),\ i \in \mathbb{N}\ , \tag{6.4.8}$$

as β_i is the arrival rate during the interval $[T_i, T_{i+1})$. In addition, we assume the service times to be iid $\text{Exp}(\mu)$.

Similar to Sect. 6.2, we adopt the SMDP framework to derive the optimal decision at times points T_i. Much of our notations from Sect. 6.2 is retained in the following with the exception of β_t, which is used to denote the rate parameter of interarrival times *at* t-th decision epoch. Unlike Sect. 6.2, β_t, in this case, is already known at $(t-1)$-th arrival epoch and, hence, is a valid input to the policy.

We first observe that in the interval $(T_i, T_i + r]$, the queue length evolves like the population of a birth-death process. Further, the transition matrix $P(s)$ over a time interval I of length s is given by e^{sG_j}, where G_j denotes the $(L+1) \times (L+1)$ intensity matrix of the process in I, L being the buffer size. Note that G_j can be written as:

$$G_j = \begin{bmatrix} -\beta_j & \beta_j & 0 & \dots & \dots \\ \mu & -\mu - \beta_j & \beta_j & 0 & \dots \\ 0 & \mu & -\mu - \beta_j & \beta_j & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \mu & -\mu \end{bmatrix},$$

for some j, which is reflected in its subscript. Additionally, in the interval $(T_i + r, T_{i+1})$ there can only be packet departures and hence the transition matrix for any interval of length s, which is a subset of $(T_i + r, T_{i+1})$, is given by e^{sG_0}, where

$$G_0 = \begin{bmatrix} 0 & 0 & 0 & \dots & \dots \\ \mu & -\mu & 0 & 0 & \dots \\ 0 & \mu & -\mu & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \mu & -\mu \end{bmatrix}_{(L+1)\times(L+1)} . \tag{6.4.9}$$

We need the following lemma to derive the transition matrix between two consecutive decision epochs.

Lemma 6. If all eigenvalues of the matrix A are positive,

$$\int_0^\infty e^{-tA}dt = A^{-1} .$$

Proof. This well-known result follows from the fact that $\lim_{t\to\infty} e^{-tA} = 0$ for a positive definite matrix A. □

Using this lemma, we derive the transition matrix for the queue length between two consecutive decision epochs in the next theorem.

Theorem 7. The transition probability matrix P_i for queue length Q over the interval (T_i, T_{i+1}) is given by

$$P_i = \beta_i e^{rG_i}(\beta_i I - G_0)^{-1}.$$

Proof. In the interval $(T_i, T_i + r]$, the queue length evolves like the population of a birth-death process with intensity matrix G_i. Hence the transition matrix for $(T_i, T_i + r]$ is given by e^{rG_i}. Also, $T_{i+1} - (T_i + r) \sim \text{Exp}(\beta_i)$, which implies

$$\begin{aligned} P_i &= e^{rG_i} \int_0^\infty e^{tG_0}\beta_i e^{-\beta_i t}dt \\ &= \beta_i e^{rG_i} \int_0^\infty e^{-(\beta_i I - G_0)t}dt . \end{aligned}$$

Note that we have used the following facts (i) $e^{-c}I = e^{-cI}$ and (ii) cI commutes with any matrix of same dimension $\forall c \in \mathbb{R}$. Now,

$$\beta_i I - G_0 = \begin{bmatrix} \beta_i & 0 & 0 & \dots & \dots \\ -\mu & \mu+\beta_i & 0 & 0 & \dots \\ 0 & -\mu & \mu+\beta_i & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & -\mu & \mu+\beta_i \end{bmatrix}.$$

By Gershgorin's theorem (Gershgorin 1931), all eigenvalues of $\beta_i I - G_0$ lie in the following closed balls: $B(\beta_i, 0), B(\mu + \beta_i, \mu)$, implying that all eigenvalues are positive. Hence, by lemma 6

$$P_i = \beta_i e^{G_i r}(\beta_i I - G_0)^{-1}.$$

□

Note that $\mathsf{P}(Q_{t+1}|Q_t, 1)$ refers to the Q_t-th row of the transition matrix over the interval $[T_t, T_{t+1})$ when the packet is dropped in the t-th decision epoch. Also, the transition matrices over intervals $[T_t, T_{t+1})$ and (T_t, T_{t+1}) are same under $A_t = 1$ as the incoming packet is dropped. Thus, $\mathsf{P}(Q_{t+1}|Q_t, 1)$ is given by the

(Q_t, Q_{t+1})-th element of the transition matrix[4] P_t. In contrast, under $A_t = 0$, the queue length changes to $\max(Q_t + 1, L)$ at time T_t and the $(Q_t, \max(Q_t + 1, L))$-th element gives $\mathsf{P}(Q_{t+1}|Q_t, 0)$. Therefore, we can express the state transition probabilities as follows:

$$\begin{aligned} \mathsf{P}(S_{t+1}|S_t, 0) &= \mathsf{P}(Q_{t+1}|Q_t, 0)\, \mathbb{1}_{\beta_{t+1}=\beta_t+1}\,, \\ \mathsf{P}(S_{t+1}|S_t, 1) &= \mathsf{P}(Q_{t+1}|Q_t, 1)\, \mathbb{1}_{\beta_{t+1}=\beta_t/2}\,. \end{aligned} \tag{6.4.10}$$

Similar to (6.2.1), the reward function is defined as:

$$\begin{aligned} R(S_t, 0) &= -M\mathbb{1}_{\{(Q_t+1)/\mu>\eta\}} + \left(\sqrt{\beta_t + 1} - \sqrt{\beta_t}\right), \\ R(S_t, 1) &= -M\mathbb{1}_{\{Q_t/\mu>\eta\}} + \left(\sqrt{\frac{\beta_t}{2}} - \sqrt{\beta_t}\right). \end{aligned} \tag{6.4.11}$$

Observe that (6.4.8) implies that $\tau(S_t, A_t) = r + 1/\beta_t$. Equipped with the inputs, similar to Sect. 6.2, we first transform the SMDP to its equivalent discrete time MDP and perform value iteration to derive the best policy.

6.4.2 *AQM under Multiple Flows*

In this section, we extend our framework to the case where multiple flows go through the switch and each flow has a different RTT. We build the framework here assuming flow RTTs are either known or already estimated at the switch running AQM.

As before, the decision times for a flow of interest are the epochs of the first packet arrivals that are at least an RTT apart. Put another way, AQM decisions on packets from the same flow are at least one RTT (of the corresponding flow) apart and all intermediate arrivals from this flow are admitted so long as there is enough room in the buffer. However, unlike the single flow case, the decision epochs need not be separated by an RTT of any flow as multiple flows are involved. Instead, the vector containing the time since the last decision epoch for each flow decision determines the timing of the next decision epoch.

Let the packet arrival times for the j-th flow be denoted by $\{X_{js}\}$, where the packet index $s \in \mathbb{N}$. Consistent with the single flow case, we assume

$$X_{j(i+1)} - X_{ji}|\beta_{jk} \sim \text{Exp}(\beta_{jk}),\ i \in \mathbb{N},$$

for some $k \in \mathbb{N}$, $j \in [n]$, $[n] = \{1, 2, \ldots, n\}$ and n denotes the number of flows passing through the switch. We denote the decision times as $\{T_s\}$, $s \in \mathbb{N}$ and see that the arrival rate for each flow remains unchanged between two decision times.

4 Recall from Thm. 7 that P_i is the transition probability matrix for queue length Q between i-th and $(i+1)$-th decision epoch.

Let us denote by $\boldsymbol{\beta}_t = (\beta_{1t}, \beta_{2t}, \ldots, \beta_{nt})^T$ the parameter vector of packet interarrival times at the t-th decision epoch and let $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)^T$ denote the vector of flow RTTs. At t-th decision epoch, wd define the *age* of the jth flow as the time since the last decision epoch concerning a packet from the j-th flow and denote it by u_{jt}. The corresponding age vector is expressed as $\boldsymbol{u}_t = (u_{1t}, u_{2t}, \ldots, u_{nt})^T$. If the time of the last decision epoch concerning flow j is unknown, e.g., before the arrival of the first packet from flow j, we let $u_{jt} = r_j$. Evidently, if the t-th decision time involves a packet from flow j, we have $u_{jt} = 0$. See Fig. 6.4 for an illustration of the decision times when multiple flows pass through the switch.

At the t-th decision epoch, let us denote by Y_{jt} the time until the next AQM decision involving flow j. Equivalently, Y_{jt} denotes the time until the arrival of a packet from flow j that is at least one RTT apart from the last decision epoch involving flow j.

Given the RTT r_j of flow j, we see that

$$Y_{jt} = W_j + \max(0, r_j - u_{jt}) \ ,$$

where W_j denotes the time until the next packet arrival from j-th flow given r_j amount of time has passed since the last decision epoch involving flow j. Using the memoryless property of exponential distribution, we note that $W_j \sim \text{Exp}(\beta_{jt})$, for $j \in [n]$. Therefore, the time until the $(t+1)$-th decision epoch across flows can be written as

$$Y_t = \min_{j \in [n]} Y_{jt} \ . \tag{6.4.12}$$

This leads to the following expression for the inter-decision time $T_{t+1} - T_t = Y_t$

$$Y_t | \boldsymbol{\beta}_t, \boldsymbol{u}_t \sim \min_{j \in [n]} \{W_j + \max(0, r_j - u_{jt})\}, \ t \in \mathbb{N},$$

Using the definition above, we now formalize the SMDP framework to derive an optimal AQM policy. As before, this policy takes in the current switch state and prescribes the ideal action (drop/admit). Along with the current flow arrival rates and the queue length, the state description, in this case, includes the *age* vector $\boldsymbol{u}_t$ and the flow index of the incoming packet. For the formulation of the SMDP, we retain most of the notations introduced in Sect. 6.2 with the following exceptions:

$\boldsymbol{\beta}_t$: flow-wise rate parameter *vector* of packet interarrival times *at* t-th decision epoch,

$\boldsymbol{u}_t$: flow-wise age vector, where each element denotes the time since last decision epoch involving a packet from a particular flow at t-th decision epoch,

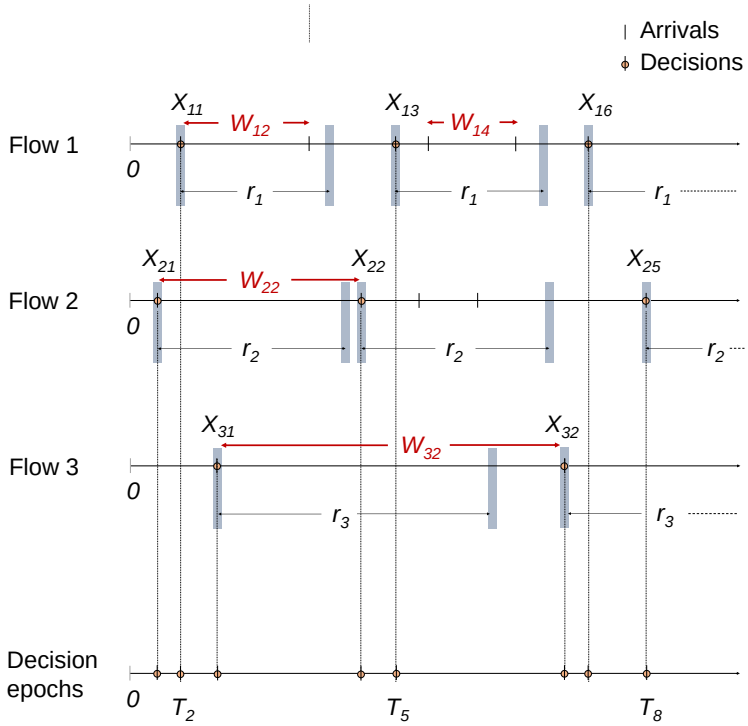


Figure 6.4: Illustration of decision times when three flows pass through the switch. The first decision time is the first arrival, which takes place at $T_1 = X_{21}$. Next decision time concerning flow 2 cannot be earlier than $X_{21} + r_2$. The first arrival from flow 1 or 3 can potentially be the next decision time, which turns out to be $T_2 = X_{11}$. Likewise, the decision process on flow 1 goes dormant in $(X_1, X_1 + r]$. Decision times are also the times where the arrival rates change for the concerning flow. The interarrival time W_{ij} is exponentially distributed with a parameter that was fixed at the last decision time concerning flow i, according to the action taken at the second last decision time on that flow.

> S_t: state of the system observed by the packet at t-th decision epoch, given by $(j, Q_t, \boldsymbol{\beta}_t, \boldsymbol{u}_t)$ where j is the flow index of the packet for which the AQM decision is to be taken.

Note that the flow-wise age $\boldsymbol{u}_t$ is a continuous variable and hence we derive the corresponding transition kernels instead of transition probabilities. Let us denote by $B = (j, Q_{t+1}, \boldsymbol{\beta}_{t+1}, U)$ the set of possible system states at $(t+1)$-th decision epoch, i.e., $S_{t+1} \in B$. Following standard convention, we restrict our attention to $U \in \mathcal{B}(\mathbb{R}^n)$, i.e., the Borel sets of $\mathbb{R}^n$. Since right semi-closed rectangles are known to generate $\mathcal{B}(\mathbb{R}^n)$, it is sufficient to look at the sets U of the form $U = I_1 \times I_2 \times \ldots I_n$, each I_j being a right semi-closed interval in $\mathbb{R}$.

Observe from (6.4.12) that the value the age vector takes at $(t+1)$-th decision epoch is given by $\boldsymbol{u}_t + Y_t$, which motivates us to define the translates

$$I'_j = \{x : x + u_{jt} \in I_j\} \ , \ j \in [n] \ .$$

Using the translates we can express the transition kernel as follows:

$$\begin{aligned}
\mathsf{P}\left(S_{t+1} \in B | S_t, 0\right) &= \mathbb{1}_{\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \boldsymbol{e}_i} \, \mathsf{P}\left(Y_t \in \bigcap_{k\in[n]} I'_k, Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t | A_t = 0\right) , \\
\mathsf{P}\left(S_{t+1} \in B | S_t, 1\right) &= \mathbb{1}_{\boldsymbol{\beta}_{t+1} = h(\boldsymbol{\beta}_t, i)} \, \mathsf{P}\left(Y_t \in \bigcap_{k\in[n]} I'_k, Y_t = Y_{kt}, V_t = Q_{t+1} - Q_t | A_t = 1\right) ,
\end{aligned} \tag{6.4.13}$$

where V_t denotes the change in queue length over $[T_t, T_{t+1})$ and $h(\boldsymbol{x}, i) = (x_1, x_2, \ldots, x_i/2, \ldots, x_n)$ and e_i is the unit vector whose i-th coordinate equals 1. Here the indicator functions on the RHS of (6.4.13) refers to the fact that the arrival rate changes to a specific value that is determined by the action (drop/admit) and the AIMD principle of TCP. Also, the incoming packet at $(t+1)$-th decision epoch is from flow j iff the corresponding inter-decision time $Y_t = Y_{jt}$. Lastly, the age vector $\boldsymbol{u}_{t+1}$ in the next decision lies in the set U iff $Y_t \in I_k - u_{kt}$, $1 \le k \le n$. Let us further introduce the following notations for brevity:

$$\begin{aligned}
&\bigcap_{j\in[n]} I'_j = (a, b], \text{ when } \bigcap_{j\in[n]} I_j \neq \phi \ , \\
&\max(0, r_j - u_{jt}) := c_j, \quad j \in [n] \ .
\end{aligned}$$

Note that c_j denotes the minimum time gap between two decision epochs involving flow j. Evidently, after a decision involving the i-th packet, c_i is reset to the RTT of the i-th flow. Put another way, if $S_t = \{i, Q_t, \boldsymbol{\beta}_t, \boldsymbol{u}_t\}$, $c_i = r_i$. Let us now order the time spans as $c_{i_1} \le c_{i_2} \le \cdots \le c_{i_n}$. Clearly, the time until the next decision is more than c_{i_1}, i.e., for $(a, b] \subset (-\infty, c_{i_1}]$, $P(Y_t \in (a, b]) = 0$. For $(a, b] \subset (c_{i_l}, c_{i_{l+1}}]$ we can express the corresponding probability as:

$$\begin{aligned}
&\mathsf{P}\left(Y_t \in (a, b], Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t\right) \\
&= \int_a^b \mathsf{P}\left(Y_t = y, V_t = Q_{t+1} - Q_t | Y_{jt} = y\right) dF_{Y_{jt}}(y) \\
&= \mathbb{1}_{j\in\{i_1 \ldots i_l\}} \int_a^b \mathsf{P}\left(\bigcap_{k\le l, i_k \neq j} Y_{i_k t} > y\right) \mathsf{P}\left(V_t = Q_{t+1} - Q_t | Y_t = y\right) dF_{Y_{jt}}(y) \\
&= \mathbb{1}_{j\in\{i_1 \ldots i_l\}} \int_a^b \prod_{k\le j, i_k \neq j} \mathsf{P}\left(Y_{i_k t} > y\right) \mathsf{P}\left(V_t = Q_{t+1} - Q_t | Y_t = y\right) dF_{Y_{jt}}(y) \ .
\end{aligned} \tag{6.4.14}$$

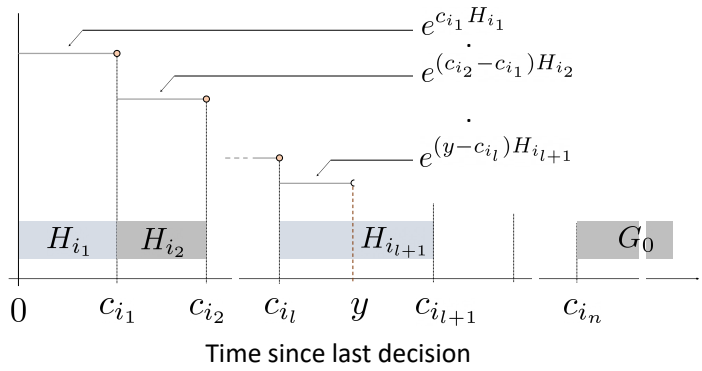


Figure 6.5: Intensity and transition matrices between two decision epochs: intensity matrices (H) are highlighted in grey over time since the last decision epoch. Each interval has a fixed intensity matrix H implying that the transition matrix over sub-interval of length x is e^{xH}. For an interval containing contiguous subintervals, the transition matrix is the product of respective transition matrices as shown for the interval $(0, y)$.

Here the indicator function $\mathbb{1}_{j\in\{i_1\ldots i_l\}}$ refers to the fact that the flow index of the packet in the next decision epoch is j only if $c_j \leq c_{i_l}$. Further, the first term in the integrand on third line of (6.4.14) indicates that the events $\{Y_t = Y_{jt}\}$ and $\{Y_{i_k t} > y\ \forall k \leq l, i_k \neq j\}$ are equivalent given $Y_{jt} = y$.

To evaluate $P(V_t = Q_{t+1} - Q_t | Y_t = y)$ in (6.4.14), we observe the fact that the queue length grows like the population of a birth-death process where the birth rate is time-dependent and the death rate is μ. The birth rate stays the same in each interval $(c_{i_{k-1}}, c_{i_k}], 1 \leq k \leq n$ and the rate is determined by the fact that there cannot be any packet arrival from the i_k-th flow once the time since the last decision epoch exceeds c_{i_k}; see Fig. 6.5. This leads to the $(L+1)\times(L+1)$ intensity matrix[5] H_{i_k} for the interval $(c_{i_{k-1}}, c_{i_k}], 1 \leq k \leq n$, where

$$H_{i_k} = \begin{bmatrix} -b_k & b_k & 0 & \ldots & \ldots \\ \mu & -\mu - b_k & b_k & 0 & \ldots \\ 0 & \mu & -\mu - b_k & b_k & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \mu & -\mu \end{bmatrix}.$$

Here we have the sum of flow rates $b_k = \sum_{m\geq k} \beta_{i_m}$ and $c_{i_0} = 0$ by convention. Additionally, the intensity matrix corresponding to (c_{i_n}, ∞) is given by G_0 from (6.4.9).

5 Recall that the buffer of the switch running AQM can hold at most L packets.

Let us denote the transition kernel for $P(Y_t \in (a,b], Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t)$ as $G_j(a,b)$. Substituting $P(V_t = Q_{t+1} - Q_t | Y_t = y)$ in (6.4.14) by the corresponding transition matrix, we get

$$\begin{aligned}
&G_j(a,b) \\
&= \mathbb{1}_{j\in\{i_1\ldots i_l\}} \prod_{1\le k\le l} e^{\left((c_{i_k}-c_{i_{k-1}})H_{i_k}\right)} \int_a^b e^{\left(\sum_{1\le k\le l, i_k\neq j} -\beta_{i_k}(y-c_{i_k})\right)} e^{(y-c_{i_l})H_{i_{l+1}}} \beta_j e^{-\beta_j(y-c_j)} dy \\
&= \mathbb{1}_{j\in\{i_1\ldots i_l\}} \beta_j e^{\left(\sum\limits_{1\le k\le l} \beta_{i_k} c_{i_k}\right)} e^{-c_{i_l}H_{i_{l+1}}} \prod_{1\le k\le l} e^{(c_{i_k}-c_{i_{k-1}})H_{i_k}} \int_a^b e^{\left(\sum_{1\le k\le l} -\beta_{i_k} y\right)} e^{yH_{i_{l+1}}} dy \\
&= \mathbb{1}_{j\in\{i_1\ldots i_l\}} \beta_j e^{\left(\sum\limits_{1\le k\le l} \beta_{i_k} c_{i_k}\right)} e^{-c_{i_l}H_{i_{l+1}}} \prod_{1\le k\le l} e^{(c_{i_k}-c_{i_{k-1}})H_{i_k}} \left(\sum_{1\le k\le l} \beta_{i_k} I - H_{i_{l+1}}\right)^{-1} \\
&\quad \left(e^{a\left(\sum_{1\le k\le l}\beta_{i_k} I - H_{i_{l+1}}\right)} - e^{b\left(\sum_{1\le k\le l}\beta_{i_k} I - H_{i_{l+1}}\right)}\right).
\end{aligned} \tag{6.4.15}$$

Note that the (Q_t, Q_{t+1})-th element of the kernel gives the probability under the drop action $A_t = 1$ and the corresponding probability under $A_t = 0$ is given by the $(Q_t, \max(Q_t+1, L))$-th element. Further, consistent with the single flow case, we define the reward function $R(S_t, A_t)$ as

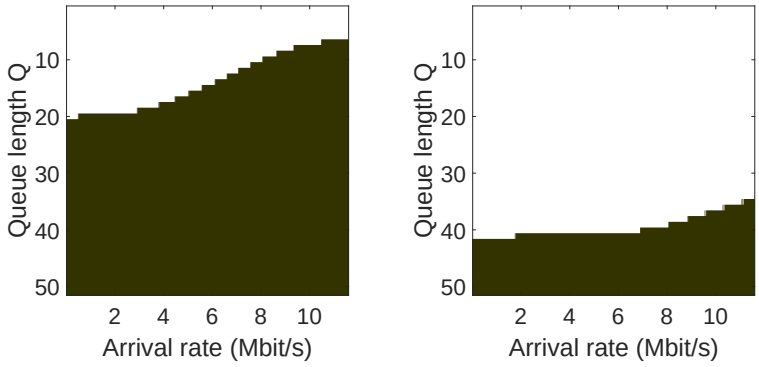
$$\begin{aligned}
R(S_t, 0) &= -M\mathbb{1}_{\{(Q_t+1)/\mu > \eta\}} + \left(\sqrt{\beta_j + 1} - \sqrt{\beta_j}\right), \\
R(S_t, 1) &= -M\mathbb{1}_{\{Q_t/\mu > \eta\}} + \left(\sqrt{\frac{\beta_j}{2}} - \sqrt{\beta_j}\right).
\end{aligned} \tag{6.4.16}$$

Using the transition probabilities and the reward function, we first derive a function approximator for the Q-function, which denotes the long-term value of a state-action pair. The estimated Q-values accordingly prescribes the best action in each state and thus we can express the optimal policy as a function of the state of the switch. Next, we evaluate our AQM policy under different network scenarios.

6.5 NUMERICAL EVALUATION

In this section, we first evaluate our AQM policy under the assumption of negligible RTT. The evaluation under non-negligible RTT is subsequently taken up in Sect. 6.5.2. Throughout our simulations, respective optimal AQM policies aim to achieve a delay shorter than the target delay $\eta = 50$ ms while optimizing the throughput. Further, the penalty amount in the reward function[6] for breaching

6 The reward function combines individual delay and throughput objectives; see (6.4.11), (6.2.1).



(a) Service rate = 5 Mbit/s **(b)** Service rate = 10 Mbit/s

Figure 6.6: Optimal AQM policy under non-negligible RTT for (a) low and (b) high service rates. The policy is shown for arrival rates $\in [0.01, 12]$ Mbps and queue lengths truncated at 50 packets. The delay threshold $\eta = 50$ ms corresponds to 20 packets in the low service regime in (a) and to 40 packets in (b). The darker region shows where an incoming packet should be dropped. As expected, the policy drops packets much earlier if the corresponding service rate is low.

this delay is fixed at $M = 10^6$. To make comparisons fair, we set the target delay parameter of CoDel to 50 ms, wherever applicable.

While comparing the performance of different policies, we look at the empirical stationary behaviour of the corresponding system. To derive stationary behaviour, we conduct 200 simulation runs. Each run starts with an arrival rate that corresponds to 100% system utilization and spans across 5×10^4 packet arrivals. We subsequently time average the state of the system, given by the buffer filling of the switch and the flow arrival rate, to generate respective illustrations. Note that the stationary arrival rate equals the stationary throughput and the queue length exclusively determines the *expected* delay. We use these immediate connotations to interpret the findings from our plots. Additionally, in these illustrations, we show the propagation of time by varying the colour of the state from blue to yellow and the optimal policy is captioned as OPT AQM.

6.5.1 *Negligible RTT*

In this section, we analyze the performance of the AQM policy derived for the scenario when the underlying switch serves a single flow with negligible RTT[7]. We follow the method described in Sect. 6.2 to compute the optimal policy and the resulting state evolution. Recall that we base our analysis on the assumption

7 For an applicable scenario, see Fig. 6.1.

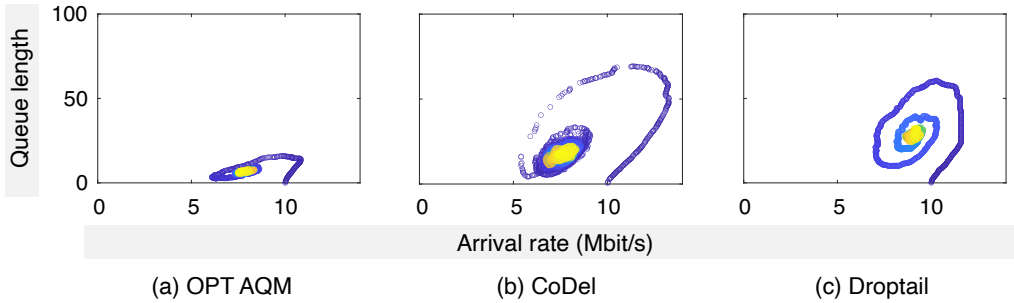
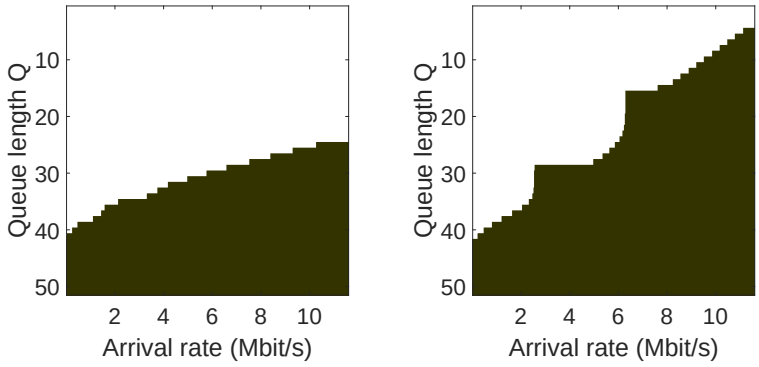


Figure 6.7: State evolution under different policies for a switch with service rate = 10 Mbps. As already known, the drop tail policy achieves higher stationary utilization than AQM policies at the cost of higher delay. Our AQM policy achieves comparable stationary throughput to CoDel while leading to a much shorter delay. The faster convergence to steady-state under the optimal policy is also worth noting.

that the interarrival times are Gamma distributed while the service times follow an exponential distribution.

We first plot the derived policies using the inputs from (6.2.3) for low and high service rates at the switch. The simulating setup for the scenario of low service corresponds to a service rate $\mu = 5$ Mbps while the high service rate is given by $\mu = 10$ Mbps. Further, we calculate the policy for arrival rates in the range $[0.01, 12]$ Mbps, simulated using a Gamma random variable with fixed shape parameter $\alpha = 1.5$. For our simulations, only the rate parameter β of the Gamma variable is changed to reflect the impact of the drop/admit action on the effective arrival rate β/α. The resulting policies are shown in Fig. 6.6, where an incoming packet should be dropped if the current state of the switch, i.e., buffer filling and the packet arrival rate belong to a dark region. As expected, lower service rate at the switch entails more aggressive packet drops reflected by the difference in Fig. 6.6a and 6.6b. The non-trivial effect of flow arrival rate on the policy is also noteworthy.

Next, we compare the performance of the optimal policy to the popular AQM policy CoDel and the classic drop tail policy. As mentioned earlier, the target delay parameter of CoDel is set to the delay threshold $\eta = 50$ ms. The evaluation is shown in Fig. 6.7, where the policy from Fig. 6.6b is used to generate the left subplot. We plot the system state, given by the current buffer filling and flow arrival rate, time-averaged over multiple runs. Our plots suggest that, in stationarity, the optimal AQM policy results in an arrival rate that is comparable to CoDel, although the queue length appears to be much shorter. This immediately translates to the fact that the optimal policy yields equivalent stationary



(a) RTT = 2 ms **(b)** RTT = 10 ms

Figure 6.8: Optimal AQM policy for a flow with known RTT. The left subplot assumes a low RTT of 2 ms and the right corresponds to a high RTT of 10 ms. Similar to Fig. 6.6, we show the policy for arrival rate $\in [0.01, 12]$ Mbps and truncate the queue length at 50 packets. The darker region shows where an incoming packet should be dropped. As our policy admits all incoming packets in the interval between two decision events which is at least one RTT long, it compensates by dropping packets in (b) earlier than (a). This phenomenon becomes more apparent as the arrival rate increases.

throughput while keeping the delay much shorter. As already known, we see that the drop tail policy generates near-perfect utilization at the cost of a longer delay.

6.5.2 *Non-negligible RTT*

In this section, we consider the case when the flow RTT is non-negligible. We use the framework from Sect. 6.4 to derive the optimal policy for each case and accordingly simulate the system. As before, we compare the resulting stationary performance empirically with CoDel and drop tail queues.

In Fig. 6.8, we derive AQM policies under low and high RTTs. We fix the packet service rate at 10 Mbps and restrict our attention to arrival rate $\in [0.01, 12]$ Mbps. In contrast to Sect. 6.5.1, packet interarrival times are simulated using an exponential distribution as assumed in Sect. 6.4. Recall that our policy, in this case, admitted all intermediate arrivals between two decision epochs subject to sufficient room at the buffer of the switch. This leads to increased packet buffering for higher RTTs, especially in high arrival regimes as the inter-decision times are RTT-based. To alleviate this problem, the policy starts dropping at shorter queue length as seen via a comparison of Fig. 6.8a and 6.8b.

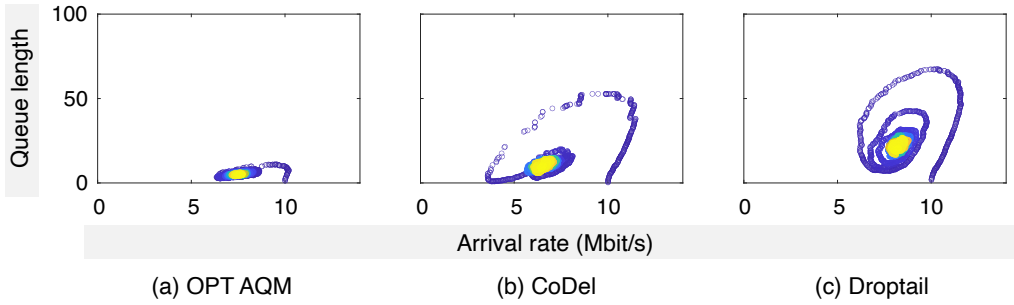


Figure 6.9: State evolution under different policies for a flow with known RTT = 2 ms and switch service rate = 10 Mbps. Similar to Fig. 6.7, our policy leads to equivalent stationary throughput to CoDel with the advantage of shorter delay and faster convergence.

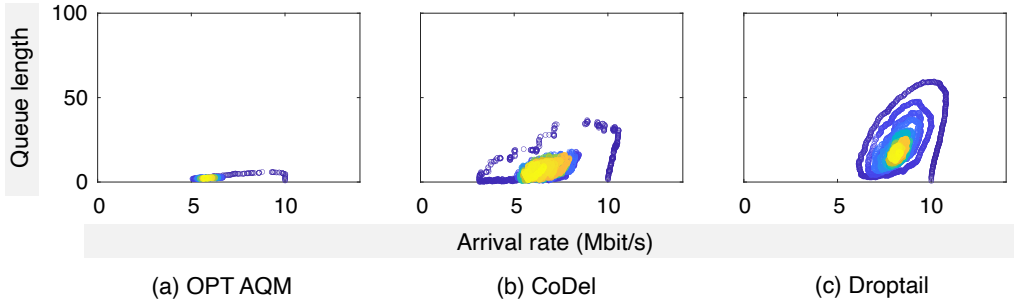
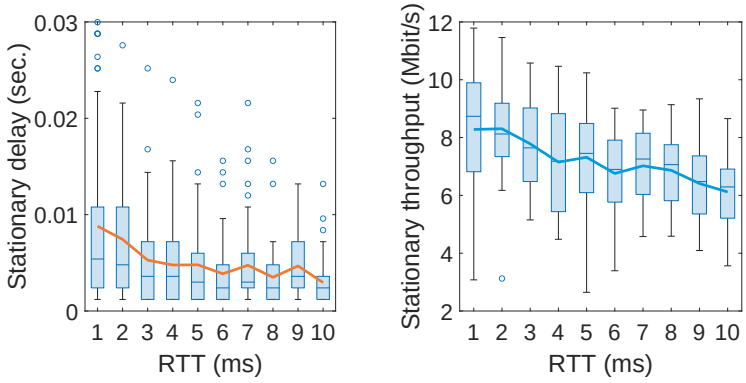


Figure 6.10: State evolution under different policies for a setup same as Fig. 6.9 except that the flow RTT is much higher (10 ms). We observe similar phenomena as Fig. 6.9 although both the stationary delay and the throughput tend to decrease across AQM policies.

In Fig. 6.9 and 6.10, we compare the optimal policy with its counterparts. The state evolution of the optimal policy (left subplot) in each of these plots is generated using the corresponding policy from Fig. 6.8. Similar to Sect. 6.5.1, we see that the optimal policy leads to comparable stationary throughput as CoDel while keeping the stationary delay much shorter. Further, a quick comparison of Fig. 6.9 and 6.10 reveals that a longer flow RTT leads to a reduction in both throughput and delay, which we investigate further in our next plot.

In Fig. 6.11, we look at the effect of flow RTT on the stationary delay and throughput. For a given RTT, we first derive the optimal policy and subsequently simulate the system for 5×10^4 arrival events. The box plot for a given RTT is generated using 200 simulation runs. Consistent with Fig. 6.7, 6.9 and 6.10, we see that the stationary delay and the stationary throughput tend to be lower as RTT increases.



(a) Delay **(b)** Throughput

Figure 6.11: Impact of flow RTT on the stationary delay and throughput. Given an RTT, we compute the optimal policy and simulate the system until 5×10^4 arrivals. Each box plot is generated using 200 simulation runs. Consistent with previous plots, both delay and throughput decrease with an increasing RTT.

6.6 SUMMARY

In this chapter, we considered the problem of finding an optimal AQM policy for a network switch. AQM aims to minimize buffering in switches without sacrificing too much throughput. Under an AQM policy, the switch actively drops packets, which leads to transitions of the state of the switch, expressed in terms of the existing backlog and the flow arrival rates in our formulation. These transitions enable performance optimization of the switch in that they help reduce the queueing delay of the packets without incurring unacceptable loss in throughput. Note that delay minimization and throughput optimization in switches are natural prerequisites to obtain a satisfactory level of overall networked system quality.

The state-of-the-art AQM algorithms are either heuristic-based or require non-trivial parameter engineering, on which there is no consensus. To that end, we specify a principled approach to solve the AQM problem. We first model the state evolution of the switch and, consequently, formulate a precise performance goal. Leveraging tools from the MDP literature, we learn an optimal packet dropping policy that looks at the current backlog and the flow arrival rates to reach a decision. For unknown traffic flow characteristics, we design an inference module that enables us to derive the policy in such cases. Further, we evaluate our policy numerically and compare its performance with the popular AQM policy CoDel as well as the traditional drop tail policy. Our results indicate that the prescribed policy yields throughput similar to CoDel while minimizing the queueing delay considerably.

6.7 DEFERRED PROOFS

Proof of Lemma 4. We denote the upper incomplete gamma integral by $\gamma(\cdot,\cdot)$, which has the following recursive property:

$$\gamma(x+1,y) = \int_y^\infty t^x e^{-t} dt = x\gamma(x,y) + y^x e^{-y},$$

for $\Re(x) > 0$. Now,

$$\begin{aligned}
&\mathsf{P}\left(Y_{u,v} > X_{w,z}\right) \\
&= \int_0^\infty \mathsf{P}\left(Y_{u,v} > X_{w,z} | X_{w,z} = t\right) dP_X(t) \\
&= \int_0^\infty \frac{\gamma(u,vt)}{\Gamma(u)} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt \\
&= \int_0^\infty \frac{(u-1)\gamma(u-1,vt) + (vt)^{u-1}e^{-vt}}{\Gamma(u)} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt \\
&= \mathsf{P}\left(Y_{u-1,v} > X_{w,z}\right) + \int_0^\infty \frac{v^{u-1} z^w t^{u+w-2} e^{-(v+z)t}}{\Gamma(u)\Gamma(w)} dt \\
&= \mathsf{P}\left(Y_{u-1,v} > X_{w,z}\right) + \frac{\Gamma(u+w-1) v^{u-1} z^w}{\Gamma(u)\Gamma(w)(v+z)^{u+w-1}} \int_0^\infty \frac{(v+z)^{u+w-1} t^{u+w-2} e^{-(v+z)t}}{\Gamma(u+w-1)} dt \\
&= \mathsf{P}\left(Y_{u-1,v} > X_{w,z}\right) + \frac{\Gamma(u+w-1)}{\Gamma(u)\Gamma(w)} \left(\frac{v}{v+z}\right)^{u-1} \left(\frac{z}{v+z}\right)^w.
\end{aligned}$$

This proves the first part of the lemma. By induction on u,

$$\mathsf{P}\left(Y_{u,v} > X_{w,z}\right) = \mathsf{P}\left(Y_{1,v} > X_{w,z}\right) + \sum_{k=2}^{u} \frac{\Gamma(k+w-1)}{\Gamma(k)\Gamma(w)} \left(\frac{v}{v+z}\right)^{k-1} \left(\frac{z}{v+z}\right)^w. \tag{6.7.17}$$

Since $Y_{1,v} \sim \mathrm{Exp}(v)$, we have

$$\begin{aligned}
\mathsf{P}\left(Y_{1,v} > X_{w,z}\right) = \int_0^\infty e^{-vt} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt &= \left(\frac{z}{v+z}\right)^w \int_0^\infty \frac{(v+z)^w t^{w-1} e^{-(v+z)t}}{\Gamma(w)} dt \\
&= \left(\frac{z}{v+z}\right)^w.
\end{aligned}$$

Thus (6.7.17) translates to

$$\mathsf{P}\left(Y_{u,v} > X_{w,z}\right) = \sum_{k=0}^{u-1} \frac{\Gamma(k+w)}{\Gamma(k+1)\Gamma(w)} \left(\frac{v}{v+z}\right)^k \left(\frac{z}{v+z}\right)^w.$$

□

7 CONCLUSION AND FUTURE WORK

7.1 CONTRIBUTIONS REVISITED

In this thesis, we studied the aspect of performance optimization in adaptive systems that are prevalent in modern communication networks. These systems operate under heterogeneous environments and to optimize their performance in terms of diverse Quality of Service (QoS) metrics, they dynamically change the employed mechanisms, which we term as transitions. Accordingly, we referred to these systems as transition-based systems (TBSs) and investigated the impact of transitions on performance optimization in such systems. We further noticed that transitions could be triggered by time or events and looked at the consequences of such transitions in parallel systems. The heterogeneity of devices and their varied performance objectives not only lead to different triggers but also induce transitions on different time scales. We subsequently considered a batch-processing system where transitions occurred upon system reconfiguration. Finally, we focused on the overall networked system quality, which was a prerequisite to seamless functioning of TBSs connected to a network. To that end, we considered the problem of Active Queue Management (AQM), which dealt with the issue of queueing delays in switches with large buffers while maintaining a desired level of throughput. In each case, we studied the advantage of our approach over the state of the art and substantiated our claim through extensive numerical evaluations. Below, we summarize our contributions against the research questions posed at the beginning of this thesis.

Research Question	Contributions
(**RQ1**) Impact of time-triggered and event-triggered transitions on QoS in parallel systems.	Chap. 3: Policy suggested for time-based transitions. Chap. 4: Framework proposed for performance evaluation under event-based transitions.
(**RQ2**) Optimization of a given QoS metric in closed batching TBSs.	Chap. 5: Exact and approximate optimal batch size calculated for small and large systems, respectively.
(**RQ3**) Optimal transition policies in network elements for enhanced overall quality.	Chap. 6: Optimal policy derived for packet drop in network switches.

(**RQ1**) *Impact of time-triggered and event-triggered transitions on QoS in parallel systems*: We first considered time-triggered transitions in Chap. 3. The underlying TBS provided an abstraction for communication devices having multiple interfaces. The choice of the interface was decided at the beginning of each decision epoch, which had a fixed time length. Although the TBS was equipped with multiple communication technologies, all packets during a single epoch could only be channelled to a single interface. We proposed a policy that looked at the maximum possible delay at the end of the epoch while making the choice of the interface at the beginning of the epoch. To derive the delay bound, we adopted a service-curve based approach that requires a calculation of arrival and service envelopes. To that end, we also proposed a method to calculate service envelopes for Markov modulated service processes. We compared the performance of our policy with epoch-based variants of round-robin and JSQ discipline and observed that our policy led to shorter worst-case delays.

Next, we looked at TBSs consisting of heterogeneous groups of parallel servers and a single dispatcher in Chap. 4. Upon arrival of a job, the dispatcher schedules the job on a server and the decision is influenced by the existing backlog of servers and the suitability of the group/cluster with respect to the said job. The notion of suitability is motivated by the fact that clusters are usually geographically distributed and have disparate hardware capabilities. Accordingly, we suggested cost-based scheduling (CBS) that generalizes the concept of randomized load balancing in a standard supermarket model. As the decision to choose a server is influenced by server backlog, cluster suitability and QoS objective (e.g., uniform load-balancing or delay minimization), the CBS policy introduces a cost function that has the flexibility to encode these aspects. To put succinctly, once a job arrives, the dispatcher first samples[1] backlogs from each cluster, uses the encoded cost function to rank the sampled servers in terms of the cluster suitability and the desired performance objective and finally schedules the job at the most appropriate server. We then focused on the corresponding evolution of server occupancies to compare the effectiveness of different cost functions to achieve a certain performance goal. Note that such a system provides an abstraction for modern datacenters that contain tens of thousands of servers. Accordingly, we provided a scaling limit that reasonably approximated the large-system behaviour of such TBSs as seen through simulation. In contrast to the state of the art, which assume idealized infinite-buffer servers, we consider finite-buffer systems. This additionally helped us derive the impact of the buffer size on certain QoS metrics.

(**RQ2**) *Optimization of a given QoS in closed batching TBSs*: We looked at closed batch-processing systems in Chap. 5 where transitions occurred upon system reconfiguration. The TBS in this case comprises of three stations: a client station

1 As retrieving backlog from each server is infeasible in large systems.

hosting clients that submit jobs, a batching station where jobs are batched and the service station consisting of multiple servers. Once a job is submitted by a client, it waits in the batching station until there are enough jobs to batch. Note that the batch size is a parameter that can be fixed during system reconfiguration. Once jobs form a batch, they are forwarded to the service station and wait in a common queue until one of the servers becomes available. A batch is served together and the responses are sent to the client station where they are split instantaneously. Upon receiving the response to a previously submitted job, a client can submit a new job after a random sleeping time. This ensures that the number of jobs in the TBS stays equal to the number of clients. In the TBS, batching is motivated by the fact that operational overhead can be distributed over multiple jobs, which leads to reduced per job service time. This phenomenon is referred to as batching speedup. However, the benefit of batching speedup is not unmitigated as a very large batch size can lead to idle servers and, consequently, sub-optimal system throughput. Accordingly, we looked for methods to derive the optimal batch size maximizing the throughput of the system. We first used a continuous time Markov chain framework to model system behaviour, which let us calculate the optimal batch size with considerable accuracy. However, as the system size grows, this approach fails owing to associated computational cost. We thus turn to a mean-filed analysis of the system, which yields optimal batch size in time independent of the system size. We evaluated results from both approaches in a prototype of a commercial database system and found them to capture system behaviour particularly well.

(**RQ3**) *Optimal transition policies in network elements for enhanced overall networked system quality*: In this thesis, we focused on the aspect of performance optimization in TBSs connected to a communication network. Needless to say, the basic network functioning has a significant impact on the performance of a TBS. Thus, we considered performance optimization of a basic network element– a switch, in Chap. 6. In particular, we focused on the AQM problem that requires throughput optimization in network switches while adhering to a user-defined delay guarantee. Compared to the state of the art, which is either heuristic-based or requires experimental parameter engineering, we resort to a principled approach. We adopted the framework of Markov Decision Processes (MDP) where we specified the performance objective of the switch using a precise reward function. Subsequently, using the tools from MDP literature, we derived an optimal policy that maximized long-term gain in performance. The optimal policy used current buffer filling at the switch and the packet arrival rates for incumbent flows to make a decision of packet drop. For unknown arrival characteristics, we also proposed a way to infer them. We evaluated our policy using simulations, which showed that it yielded similar throughput as CODEL, the popular AQM algorithm, while keeping the delay at the switch at a significantly lower level.

7.2 FUTURE WORK

In this work, we discussed performance optimization in TBSs. We took examples of different classes of TBSs and provided a method to enhance their performance. However, in the context of a particular system class, the system setup can be further generalized to make the analysis relevant to a larger class of real-world system. In the context of the parallel system employing CBS policy in Chap. 4, our system builds on the assumption of exponential job interarrival and service time. This assumption can be relaxed to broaden the scope of the derived result. For example, systems where jobs arrive in bursts or servers extend multiple stages of service are not amenable to our framework. Although our analysis readily extends to the case of batch arrival, which we do not discuss here, the possibility of admitting a batch partially when there is insufficient room requires further analysis.

In the context of the closed batch-processing system in Chap. 5, our results can be extended to the case where the batches go through multiple stages of service, i.e., the system contains series of service stations. Recall that for the analysis with multiple job types, we assumed equal batch size for all types. However, allowing different batch sizes for different types will optimize the system throughput even further. We also see that our results are derived on the assumption of preemptive service priority between job types, which precludes many closed batching systems such as databases employing non-preemptive service priority[2].

While our results in Chap. 6 were compared with the state of the art using simulations, a logical next step would be to carry out the comparison in emulation or in real systems. As a general observation, our analysis for the specific case where flows have non-negligible RTT is based on the assumption of exponentially distributed interarrival times. While this assumption considerably reduces the size of the state-space and makes the subsequent optimization process tractable, further generalization will help us broaden the scope such as considering bursty packet arrivals. Finally, the aspect of fairness under AQM can also be investigated, which we did not consider in this thesis.

2 Although we saw equivalence of both priority orders in our system through simulation, we did not present a rigorous proof of the same.

BIBLIOGRAPHY

Abbas, N. (2011). "Towards Autonomic Softw. Product Lines." In: *Proc. 15th Int. Software Product Line Conf.* SPLC '11. ACM, 44:1–44:8 (cit. on p. 13).

Adams, R. (2013). "Active Queue Management: A Survey." In: *IEEE Communications Surveys Tutorials* 15.3, pp. 1425–1476 (cit. on pp. 20, 77).

Adusumilli, K. M. and J. J. Hasenbein (Oct. 2010). "Dynamic admission and service rate control of a queue." In: *Queueing Systems* 66.2, pp. 131–154 (cit. on p. 15).

Aghajani, R., X. Li, and K. Ramanan (Dec. 2017). "The PDE Method for the Analysis of Randomized Load Balancing Networks." In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.2, 38:1–38:28 (cit. on p. 17).

Aghajani, R. and K. Ramanan (2017). "The hydrodynamic limit of a randomized load balancing network." In: *arXiv preprint arXiv:1707.02005* (cit. on p. 17).

Alt, B., M. Weckesser, C. Becker, M. Hollick, S. Kar, A. Klein, R. Klose, R. Kluge, H. Koeppl, B. Koldehofe, W. R. Khudabukhsh, M. Luthra, M. Mousavi, M. Mühlhäuser, M. Pfannemüller, A. Rizk, A. Schürr, and R. Steinmetz (2019). "Transitions: A Protocol-Independent View of the Future Internet." In: *Proceedings of the IEEE* 107.4, pp. 835–846 (cit. on pp. ix, x).

Amoui, M., M. Salehie, S. Mirarab, and L. Tahvildari (Mar. 2008). "Adaptive Action Selection in Autonomic Software Using Reinforcement Learning." In: *4th Int. Conf. Autonomic and Autonomous Syst.* Pp. 175–181 (cit. on p. 13).

Asadi, A. and V. Mancuso (Fourth 2013). "A Survey on Opportunistic Scheduling in Wireless Communications." In: *IEEE Commun. Surveys Tut.* 15.4, pp. 1671–1688 (cit. on p. 14).

Bailey, N. T. (1954). "On queueing processes with bulk service." In: *J. R. Stat. Soc. B. Met.*, pp. 80–87 (cit. on pp. 18, 52).

Balsamo, S., V. D. N. Personé, and P. Inverardi (2003). "A review on queueing network models with finite capacity queues for software architectures performance prediction." In: *Performance Evaluation* 51.2. Queueing Networks with Blocking, pp. 269–288 (cit. on p. 17).

Banerjee, S. and D. Mukherjee (Apr. 2019). "Join-the-shortest queue diffusion limit in Halfin-Whitt regime: Tail asymptotics and scaling of extrema." In: *The Annals of Applied Probability* 29.2, pp. 1262–1309 (cit. on p. 17).

Banerjee, S. and D. Mukherjee (Feb. 2020). "Join-the-Shortest Queue diffusion limit in Halfin-Whitt regime: Sensitivity on the heavy-traffic parameter." In: *The Annals of Applied Probability* 30.1, pp. 80–144 (cit. on p. 17).

Barré, S., C. Paasch, and O. Bonaventure (2011). "Multipath TCP: from theory to practice." In: *International conference on research in networking*. Springer, pp. 444–457 (cit. on p. 1).

Baskett, F., K. M. Chandy, R. R. Muntz, and F. G. Palacios (1975). "Open, closed, and mixed networks of queues with different classes of customers." In: *j. ACM* 22.2, pp. 248–260 (cit. on p. 19).

Berg, M., F. van der Duyn Schouten, and J. Jansen (1998). "Optimal batch provisioning to customers subject to a delay-limit." In: *Manag. Sci.* 44.5, pp. 684–697 (cit. on p. 18).

Berry, R. A. and R. G. Gallager (May 2002). "Communication over fading channels with delay constraints." In: *IEEE Transactions on Information Theory* 48.5, pp. 1135–1149 (cit. on p. 16).

Bettesh, I. and S. S. Shamai (Sept. 2006). "Optimal Power and Rate Control for Minimal Average Delay: The Single-User Case." In: *IEEE Transactions on Information Theory* 52.9, pp. 4115–4141 (cit. on p. 15).

Billingsley, P. (1999). *Convergence of Probability Measures*. Second. John Wiley & Sons, Inc., New York, pp. x+277 (cit. on p. 45).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer (cit. on p. 30).

Bleuler, S., M. Laumanns, L. Thiele, and E. Zitzler (2003). "PISA—a platform and programming language independent interface for search algorithms." In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, pp. 494–508 (cit. on p. 2).

Bolch, G., S. Greiner, H. d. Meer, and K. S. Trivedi (2005). *Queueing Networks and Markov Chains*. New York, NY, USA: Wiley-Interscience (cit. on p. 52).

Boor, M. van der, S. C. Borst, J. S. van Leeuwaarden, and D. Mukherjee (2018). "Scalable load balancing in networked systems: A survey of recent advances." In: *arXiv preprint arXiv:1806.05444* (cit. on p. 16).

Bortolussi, L. and N. Gast (2016). "Mean-Field Limits Beyond Ordinary Differential Equations." In: ed. by M. Bernardo, R. De Nicola, and J. Hillston. Cham: Springer International Publishing, pp. 61–82 (cit. on p. 57).

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press (cit. on p. 68).

Brightwell, G., M. Fairthorne, and M. J. Luczak (2018). "The Supermarket Model with Bounded Queue Lengths in Equilibrium." In: *Journal of Statistical Physics* 173.3, pp. 1149–1194 (cit. on p. 16).

Bui, V., W. Zhu, A. Botta, and A. Pescape (Oct. 2010). "A Markovian Approach to Multipath Data Transfer in Overlay Networks." In: *IEEE Trans. Parallel Distrib. Syst.* 21.10, pp. 1398–1411 (cit. on p. 14).

Chandy, K. M., J. H. Howard Jr, and D. F. Towsley (1977). "Product form and local balance in queueing networks." In: *Journal of the ACM (JACM)* 24.2, pp. 250–263 (cit. on p. 19).

Chandy, K. M. and A. J. Martin (1983). "A characterization of product-form queuing networks." In: *Journal of the ACM* 30.2, pp. 286–299 (cit. on p. 19).

Chang, C.-S. (2000). *Performance Guarantees in Communication Networks*. Springer-Verlag (cit. on p. 25).

Chaudhry, M. and J. Templeton (1983). *A first course in bulk queues*. Wiley (cit. on p. 52).

Chiu, D.-M. and R. Jain (1989). "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks." In: *Computer Networks and ISDN systems* 17.1, pp. 1–14 (cit. on pp. 77, 80).

Ciucu, F., A. Burchard, and J. Liebeherr (June 2006). "Scaling properties of statistical end-to-end bounds in the network calculus." In: 14.6, pp. 2300–2312 (cit. on pp. 24–26).

Ciucu, F., F. Poloczek, and A. Rizk (June 2019a). "Queue and Loss Distributions in Finite-Buffer Queues." In: *Proc. ACM Meas. Anal. Comput. Syst.* 3.2, 31:1–31:29 (cit. on p. 38).

Ciucu, F., F. Poloczek, and A. Rizk (2019b). "Queue and loss distributions in finite-buffer queues." In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3.2, pp. 1–29 (cit. on p. 17).

Collins, B. and R. L. Cruz (1999). "Transmission policies for time varying channels with average delay constraints." In: *Proceedings of the Annual Allerton Conference on Communication Control and Computing*. Vol. 37. The University; 1998, pp. 709–717 (cit. on p. 15).

Crabill, T. B., D. Gross, and M. J. Magazine (1977). "A Classified Bibliography of Research on Optimal Design and Control of Queues." In: *Oper. Res.* 25.2, pp. 219–232 (cit. on p. 14).

Cree, E. (2018). *Linux Kernel path: "Handle-multiple-received-packets-at-each-stage"*. Retrieved May 25, 2020 from `https://github.com/torvalds/linux/commit/2d1b138505dc29bbd7ac5f82f5a10635ff48bddb`. Accessed: 2018-10-27 (cit. on p. 51).

Cruz, R. L. (June 1996). "Quality of Service Management in Integrated Services Networks." In: *Proc. of Semi-Annual Research Review, Center of Wireless Communication, UCSD* (cit. on pp. 25, 26).

Cruz, R. L. (Mar. 1998). "SCED+: efficient management of quality of service guarantees." In: *INFOCOM '98. IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 2, 625–634 vol.2 (cit. on p. 15).

Dai, J. G. and S. P. Meyn (Nov. 1995). "Stability and convergence of moments for multiclass queueing networks via fluid limit models." In: *IEEE Transactions on Automatic Control* 40.11, pp. 1889–1904 (cit. on p. 16).

Dai, J. and W. Dai (Sept. 1999). "A heavy traffic limit theorem for a class of open queueing networks with finite buffers." In: *Queueing Systems* 32.1, pp. 5–40 (cit. on pp. 17, 18).

Deb, R. K. (1978). "Optimal dispatching of a finite capacity shuttle." In: *Manag. Science* 24.13, pp. 1362–1372 (cit. on pp. 18, 52).

Deb, R. K. and R. F. Serfozo (1973). "Optimal control of batch service queues." In: *Advances in Applied Probability* 5.2, pp. 340–361 (cit. on p. 18).

Decreusefond, L. and P. Moyal (Dec. 2008). "A functional central limit theorem for the M/GI/? queue." In: *The Annals of Applied Probability* 18.6, pp. 2156–2178 (cit. on p. 17).

DeWitt, D. J., R. H. Katz, F. Olken, L. D. Shapiro, M. R. Stonebraker, and D. A. Wood (1984). "Implementation Techniques for Main Memory Database Systems." In: *Proc. ACM SIGMOD Int. Conf. Manag. Dat.* New York, NY, USA: ACM, pp. 1–8 (cit. on p. 19).

Elkhodary, A., N. Esfahani, and S. Malek (2010). "FUSION: A Framework for Engineering Self-tuning Self-adaptive Software Systems." In: *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng.* FSE '10. ACM, pp. 7–16 (cit. on p. 13).

Engel, K.-J. and R. Nagel (1999). *One-parameter Semigroups for Linear Evolution Equations*. Vol. 194. Springer Science & Business Media (cit. on p. 44).

Ephremides, A. and S. Verdu (Sept. 1989). "Control and optimization methods in communication network problems." In: *IEEE Trans. Autom. Control* 34.9, pp. 930–942 (cit. on p. 14).

Ethier, S. N. and T. G. Kurtz (1986). *Markov Processes: Characterization and Convergence*. Characterization and convergence, pp. x+534 (cit. on pp. 17, 38, 44, 45).

Fei, Y., V. W. S. Wong, and V. C. M. Leung (Feb. 2006). "Efficient QoS Provisioning for Adaptive Multimedia in Mobile Communication Networks by Reinforcement Learning." In: *Mob. Netw. Appl.* 11.1, pp. 101–110 (cit. on p. 14).

Feldbaum, A. (1960). "Dual control theory. I." In: *Avtomatika i Telemekhanika* 21.9, pp. 1240–1249 (cit. on p. 84).

Fidler, M. and A. Rizk (Firstquarter 2015). "A Guide to the Stochastic Network Calculus." In: *IEEE Communications Surveys Tutorials* 17.1, pp. 92–105 (cit. on pp. 25, 26).

Floyd, S. and V. Jacobson (1993). "Random early detection gateways for congestion avoidance." In: *IEEE/ACM Transactions on networking* 1.4, pp. 397–413 (cit. on pp. 20, 77).

Frömmgen, A., M. Hassan, R. Kluge, M. Mousavi, M. Mühlhäuser, S. Müller, M. Schnee, M. Stein, and M. Weckesser (2016). "Mechanism transitions: A new paradigm for a highly adaptive Internet." In: (cit. on p. 6).

Garlan, D., S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste (2004). "Rainbow: Architecture- Based Self-Adaptation with Reusable Infrastructure." In: *Comput.*, pp. 46–54 (cit. on p. 13).

George, J. M. and J. M. Harrison (2001). "Dynamic Control of a Queue with Adjustable Service Rate." In: *Operations Research* 49.5, pp. 720–731 (cit. on p. 15).

Germs, R. and N. van Foreest (2013). "Analysis of finite-buffer state-dependent bulk queues." In: *OR Spectrum* 35.3, pp. 563–583 (cit. on p. 19).

Gershgorin, S. A. (1931). "Über die Abgrenzung der Eigenwerte einer Matrix." German. In: *Bull. Acad. Sci. URSS* 1931.6, pp. 749–754 (cit. on p. 89).

Gershwin, S. B. (1987). "An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage Space and Blocking." In: *Operations Research* 35.2, pp. 291–305 (cit. on p. 18).

Gettys, J. (2011). "Bufferbloat: Dark buffers in the internet." In: *IEEE Internet Computing* 15.3, pp. 96–96 (cit. on pp. 20, 77).

Giannikis, G., G. Alonso, and D. Kossmann (2012). "SharedDB: Killing One Thousand Queries with One Stone." In: *PVLDB* 5.6, pp. 526–537 (cit. on pp. 19, 59).

Gilbert, E. (1960). "Capacity of a burst-noise channel." In: *Bell Systems Technical Journal* 39.5, pp. 1253–1265 (cit. on p. 30).

Godtschalk, A. S. and F. Ciucu (June 2016). "Randomized Load Balancing in Finite Regimes." In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 580–589 (cit. on p. 16).

Gordon, W. J. and G. F. Newell (1967). "Closed queuing systems with exponential servers." In: *Operations research* 15.2, pp. 254–265 (cit. on p. 19).

Grant, M., S. Boyd, and Y. Ye (2008). *CVX: Matlab software for disciplined convex programming* (cit. on p. 68).

Gromoll, H. C., A. L. Puha, and R. J. Williams (Aug. 2002). "The fluid limit of a heavily loaded processor sharing queue." In: *The Annals of Applied Probability* 12.3, pp. 797–859 (cit. on p. 17).

Ha, S., I. Rhee, and L. Xu (2008). "CUBIC: a new TCP-friendly high-speed TCP variant." In: *ACM SIGOPS operating systems review* 42.5, pp. 64–74 (cit. on p. 86).

Hark, R., M. Ghanmi, S. Kar, N. Richerzhagen, A. Rizk, and R. Steinmetz (2018). "Representative Measurement Point Selection to Monitor Software-defined Networks." In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 511–518 (cit. on p. xi).

Henderson, W., C. Pearce, P. G. Taylor, and N. M. van Dijk (1990). "Closed queueing networks with batch services." In: *Queueing systems* 6.1, pp. 59–70 (cit. on p. 19).

Henderson, W. and P. G. Taylor (1990). "Product form in networks of queues with batch arrivals and batch services." In: *Queueing Syst.* 6.1, pp. 71–87 (cit. on pp. 19, 52).

Hollot, C. V., V. Misra, D. Towsley, and W.-B. Gong (2001a). "On designing improved controllers for AQM routers supporting TCP flows." In: *Proceedings IEEE INFOCOM 2001. IEEE Computer and Communications Society*. Vol. 3. IEEE, pp. 1726–1734 (cit. on p. 20).

Hollot, C. V., V. Misra, D. Towsley, and W.-B. Gong (2001b). "A control theoretic analysis of RED." In: *Proceedings IEEE INFOCOM 2001. IEEE Computer and Communications Society*. Vol. 3. IEEE, pp. 1510–1519 (cit. on p. 20).

Huang, J. and Z. Niu (Mar. 2007). "Buffer-Aware and Traffic-Dependent Packet Scheduling in Wireless OFDM Networks." In: *2007 IEEE Wireless Communications and Networking Conference*, pp. 1554–1558 (cit. on p. 15).

Huang, J., R. A. Berry, and M. L. Honig (Nov. 2005). "Wireless scheduling with hybrid ARQ." In: *IEEE Trans. Wireless Commun.* 4.6, pp. 2801–2810 (cit. on p. 14).

Hwang, K.-S., S.-W. Tan, M.-C. Hsiao, and C.-S. Wu (Apr. 2005). "Cooperative multiagent congestion control for high-speed networks." In: *IEEE Trans. Syst., Man, and Cybern., Part B (Cybern.)* 35.2, pp. 255–268 (cit. on p. 14).

Ito, K. and F. Kappel (2002). *Evolution Equations And Approximations*. WORLD SCIENTIFIC (cit. on p. 44).

Järvinen, I. and M. Kojo (2014). "Evaluating CoDel, PIE, and HRED AQM techniques with load transients." In: *39th Annual IEEE Conference on Local Computer Networks*. IEEE, pp. 159–167 (cit. on p. 21).

Kang, K. C., S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson (1990). *Feature-oriented domain analysis (FODA) feasibility study*. Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst (cit. on p. 7).

Kar, S. and A. Rizk (2020). "A Delicate Union of Batching and Parallelization Models in Distributed Computing and Communication." In: *2020 IFIP Networking Conference (Networking)*, pp. 534–538 (cit. on p. xi).

Kar, S., A. Rizk, and M. Fidler (2018). "Multi-interface Communication: Interface Selection Under Statistical Performance Constraints." In: *2018 30th International Teletraffic Congress (ITC 30)*. Vol. 02, pp. 7–12 (cit. on pp. ix, x).

Kar, S., R. Hark, A. Rizk, and R. Steinmetz (2018). "Towards Optimal Placement of Monitoring Units in Time-Varying Networks Under Centralized Control." In: *Measurement, Modelling and Evaluation of Computing Systems*. Cham: Springer International Publishing, pp. 99–112 (cit. on p. xi).

Kar, S., R. Rehrmann, A. Mukhopadhyay, B. Alt, F. Ciucu, H. Koeppl, C. Binnig, and A. Rizk (2020). "On the Throughput Optimization in Large-scale Batch-processing Systems." In: *Performance Evaluation* 144, pp. 102–142 (cit. on pp. ix, x, 66).

Katikala, S. (2014). "Google project loon." In: *InSight: Rivier Academic Journal* 10.2, pp. 1–6 (cit. on p. 1).

Khademi, N., D. Ros, and M. Welzl (2014). "The new AQM kids on the block: An experimental evaluation of CoDel and PIE." In: *2014 IEEE Conference on*

Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 85–90 (cit. on pp. 21, 77).

KhudaBukhsh, W. R., B. Alt, S. Kar, A. Rizk, and H. Koeppl (2018). "Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies." In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 1–9 (cit. on p. xi).

KhudaBukhsh, W. R., S. Kar, B. Alt, A. Rizk, and H. Koeppl (2020a). "Generalized Cost-Based Job Scheduling in Very Large Heterogeneous Cluster Systems." In: *IEEE Transactions on Parallel and Distributed Systems* 31.11, pp. 2594–2604 (cit. on pp. ix, x, 43, 45).

KhudaBukhsh, W. R., S. Kar, B. Alt, A. Rizk, and H. Koeppl (2020b). "Generalized Cost-Based Job Scheduling in Very Large Heterogeneous Cluster Systems." In: *IEEE Transactions on Parallel and Distributed Systems* 31.11, pp. 2594–2604 (cit. on p. 38).

KhudaBukhsh, W. R., S. Kar, A. Rizk, and H. Koeppl (Mar. 2019). "Provisioning and Performance Evaluation of Parallel Systems with Output Synchronization." In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 4.1 (cit. on p. xi).

Kim, H. S. and N. B. Shroff (Dec. 2001). "Loss Probability Calculations and Asymptotic Analysis for Finite Buffer Multiplexers." In: *IEEE/ACM Trans. Netw.* 9.6, pp. 755–768 (cit. on p. 18).

Krupitzer, C., F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker (2015). "A survey on engineering approaches for self-adaptive systems." In: *Pervasive Mob. Comput.* 17, pp. 184–206 (cit. on p. 13).

Kundel, R., J. Blendin, T. Viernickel, B. Koldehofe, and R. Steinmetz (2018). "P4-CoDel: Active queue management in programmable data planes." In: *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, pp. 1–4 (cit. on p. 2).

Kurtz, T. G. (1970). "Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes." In: *Journal of Applied Probability* 7.1, pp. 49–58 (cit. on pp. 56, 57).

Kwiatkowska, M., D. Parker, and H. Qu (June 2011). "Incremental quantitative verification for Markov decision processes." In: *2011 IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw.* Pp. 359–370 (cit. on p. 13).

Lee, D., B. E. Carpenter, and N. Brownlee (2010). "Observations of UDP to TCP Ratio and Port Numbers." In: *2010 Fifth International Conference on Internet Monitoring and Protection*, pp. 99–104 (cit. on p. 77).

Lemos, R. de, H. Giese, H. A. Müller, and M. Shaw, eds. (2013). *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 14).

Li, C., A. Burchard, and J. Liebeherr (Dec. 2007). "A Network Calculus with Effective Bandwidth." In: 15.6, pp. 1442–1453 (cit. on p. 26).

Luthra, M., B. Koldehofe, N. Danger, P. Weisenburger, G. Salvaneschi, and I. Stavrakakis (2021). "TCEP: Transitions in Operator Placement to Adapt to Dynamic Network Environments." In: *Journal of Computer and Systems Sciences (JCSS), Special Issue on Algorithmic Theory of Dynamic Networks and its Applications*, pp. 1–76 (cit. on p. 2).

Luthra, M., B. Koldehofe, P. Weisenburger, G. Salvaneschi, and R. Arif (2018). "TCEP: Adapting to Dynamic User Environments by Enabling Transitions between Operator Placement Mechanisms." In: *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems (DEBS)*, pp. 136–147 (cit. on p. 35).

M. Mitzenmacher (1996). "The power of two choices in randomized load balancing." PhD thesis. University of California at Berkeley (cit. on p. 16).

Mahabhashyam, S. R. and N. Gautam (Oct. 2005). "On Queues with Markov Modulated Service Rates." In: *Queueing Systems* 51.1, pp. 89–113 (cit. on p. 15).

Martin, J. B. and Y. M. Suhov (Aug. 1999). "Fast Jackson networks." In: *Ann. Appl. Probab.* 9.3, pp. 854–870 (cit. on p. 38).

Mehta, M., S. Khakurel, and A. Karandikar (Apr. 2012). "Buffer-based channel dependent UpLink scheduling in relay-assisted LTE networks." In: *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1777–1781 (cit. on p. 15).

Misra, V., W.-B. Gong, and D. Towsley (2000). "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED." In: *ACM SIGCOMM Computer Communication Review*. Vol. 30. 4. ACM, pp. 151–160 (cit. on p. 20).

Mitzenmacher, M. (Oct. 2001). "The power of two choices in randomized load balancing." In: *IEEE Transactions on Parallel and Distributed Systems* 12.10, pp. 1094–1104 (cit. on pp. 16, 35, 36).

Mukherjee, D., S. C. Borst, J. S. van Leeuwaarden, and P. A. Whiting (2016). "Asymptotic optimality of power-of-*d* load balancing in large-scale systems." In: *arXiv preprint arXiv:1612.00722* (cit. on pp. 16, 17).

Mukhopadhyay, A. and R. R. Mazumdar (June 2016). "Analysis of Randomized Join-the-Shortest-Queue (JSQ) Schemes in Large Heterogeneous Processor-Sharing Systems." In: *IEEE Transactions on Control of Network Systems* 3.2, pp. 116–126 (cit. on pp. 17, 38).

Mukhopadhyay, A., A. Karthik, and R. R. Mazumdar (2016). "Randomized Assignment of Jobs to Servers in Heterogeneous Clusters of Shared Servers for Low Delay." In: *Stochastic Systems* 6.1, pp. 90–131 (cit. on pp. 16, 36, 38, 40, 44, 49).

Neglia, G., M. Sereno, and G. Bianchi (Sept. 2016). "Geographical Load Balancing Across Green Datacenters: A Mean Field Analysis." In: *SIGMETRICS Perform. Eval. Rev.* 44.2, pp. 64–69 (cit. on p. 36).

Nichols, K., V. Jacobson, and A. McGregor (n.d.). *and J. Iyengar, Ed.," Controlled Delay Active Queue Management*. Tech. rep. RFC 8289, DOI 10.17487/RFC8289, January 2018 (cit. on p. 82).

Nichols, K. and V. Jacobson (2012). "Controlling queue delay." In: *Communications of the ACM* 55.7, pp. 42–50 (cit. on pp. 20, 77).

Ouyang, W., S. Murugesan, A. Eryilmaz, and N. B. Shroff (Apr. 2015). "Exploiting Channel Memory for Joint Estimation and Scheduling in Downlink Networks - a Whittle's Indexability Analysis." In: *IEEE Trans. Inf. Theory* 61.4, pp. 1702–1719 (cit. on p. 14).

Pan, R., P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg (2013a). "PIE: A lightweight control scheme to address the bufferbloat problem." In: *2013 IEEE 14th international conference on high performance switching and routing (HPSR)*. IEEE, pp. 148–155 (cit. on p. 20).

Pan, R., P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg (2013b). "PIE: A lightweight control scheme to address the bufferbloat problem." In: *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, pp. 148–155 (cit. on p. 77).

Perros, H. (1989). "A bibliography of papers on queueing networks with finite capacity queues." In: *Performance Evaluation* 10.3. Queueing Networks with Finite Capacity Queues, pp. 255–260 (cit. on p. 17).

Pukelsheim, F. (1993). *Optimal design of experiments*. Vol. 50. siam (cit. on p. 68).

Rehrmann, R., C. Binnig, A. Böhm, K. Kim, W. Lehner, and A. Rizk (Aug. 2018). "OLTPshare: The Case for Sharing in OLTP Workloads." In: *Proc. VLDB Endow.* 11.12, pp. 1769–1780 (cit. on pp. 19, 51, 59).

Rizk, A. and M. Fidler (2016). "Queue-aware uplink scheduling with stochastic guarantees." In: *Computer Communications* 84, pp. 63–72 (cit. on p. 15).

Sadeghi, P., R. A. Kennedy, P. B. Rapajic, and R. Shams (Sept. 2008). "Finite-state Markov modeling of fading channels - a survey of principles and applications." In: *IEEE Signal Processing Magazine* 25.5, pp. 57–80 (cit. on p. 27).

Sadre, R., B. R. Haverkort, and A. Ost (1999). "An Efficient and Accurate Decomposition Method for Open Finite- and Infinite-Buffer Queueing Networks." In: *Proceedings of the Third International Workshop on Numerical Solution of Markov Chains*, pp. 1–20 (cit. on p. 18).

Sariowan, H., R. L. Cruz, and G. C. Polyzos (Oct. 1999). "SCED: a generalized scheduling policy for guaranteeing quality-of-service." In: *IEEE/ACM Transactions on Networking* 7.5, pp. 669–684 (cit. on p. 15).

Sellis, T. K. (Mar. 1988). "Multiple-query Optimization." In: *ACM Trans. Database Syst.* 13.1, pp. 23–52 (cit. on pp. 51, 59).

Shen, X., H. Chen, J. Dai, and W. Dai (Sept. 2002). "The Finite Element Method for Computing the Stationary Distribution of an SRBM in a Hypercube with Applications to Finite Buffer Queueing Networks." In: *Queueing Systems* 42.1, pp. 33–62 (cit. on p. 18).

Singh, A., J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. H'olzle, S. Stuart, and A. Vahdat (2015). "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network." In: *Proc. of Sigcomm* (cit. on p. 37).

Stein, M., A. Frömmgen, R. Kluge, F. Löffler, A. Schürr, A. Buchmann, and M. Mühlhäuser (2016). "TARL: Modeling topology adaptations for networking applications." In: *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, pp. 57–63 (cit. on p. 2).

Stidham, S. and R. Weber (1993). "A survey of Markov decision models for control of networks of queues." In: *Queueing Syst.* 13.1-3, pp. 291–314 (cit. on p. 14).

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. Second. The MIT Press (cit. on pp. 9, 84).

Thomson, A., T. Diamond, S.-C. Weng, K. Ren, P. Shao, and D. J. Abadi (2012). "Calvin: Fast Distributed Transactions for Partitioned Database Systems." In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD '12. Scottsdale, Arizona, USA: ACM, pp. 1–12 (cit. on p. 51).

Tijms, H. C. (2003). *A first course in stochastic models*. John Wiley and sons (cit. on pp. 80, 84).

Vvedenskaya, N. D., R. L. Dobrushin, and F. I. Karpelevich (1996). "Queueing system with selection of the shortest of two queues: An asymptotic approach." In: *Problemy Peredachi Informatsii* 32.1, pp. 20–34 (cit. on p. 16).

Weckesser, M., R. Kluge, M. Pfannemüller, M. Matthé, A. Schürr, and C. Becker (2018a). "Optimal reconfiguration of dynamic software product lines based on performance-influence models." In: *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1*, pp. 98–109 (cit. on p. 7).

Weckesser, M., R. Kluge, M. Pfannemüller, M. Matthé, A. Schürr, and C. Becker (2018b). "Optimal reconfiguration of dynamic software product lines based on performance-influence models." In: *Proceeedings of the 22nd International Conference on Systems and Software Product Line - Volume 1, SPLC 2018, Gothenburg, Sweden, September 10-14, 2018*, pp. 98–109 (cit. on p. 13).

Wen, X., B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu (June 2016). "RuleTris: Minimizing Rule Update Latency for TCAM-Based SDN Switches." In: *Proc. IEEE Int. Conf. Dist. Comput. Sys.* Pp. 179–188 (cit. on p. 51).

Weyns, D., M. U. Iftikhar, D. G. de la Iglesia, and T. Ahmad (2012). "A Survey of Formal Methods in Self-adaptive Systems." In: *Proc. 5th Int. C* Conf. Comput. Sci. and Softw. Eng.* C3S2E '12. ACM, pp. 67–79 (cit. on p. 14).

Wierman, A., L. L. H. Andrew, and A. Tang (Sept. 2008). "Stochastic analysis of power-aware scheduling." In: *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1278–1283 (cit. on p. 15).

Winston, W. (1977). "Optimality of the shortest line discipline." In: *Journal of Applied Probability* 14.1, pp. 181–189 (cit. on p. 36).

Xie, H. and A. Boukerche (May 2015). "TCP-CC: cross-layer TCP pacing protocol by contention control on wireless networks." In: *Wireless Netw.* 21.4, pp. 1061–1078 (cit. on p. 14).

Yin, Q., Y. Jiang, S. Jiang, and P. Y. Kong (Nov. 2002). "Analysis of Generalized Stochastically Bounded Bursty Traffic for Communication Networks." In: *Proc. of IEEE LCN*, pp. 141–149 (cit. on p. 26).

ERKLÄRUNG LAUT §9 PROMOTIONSORDNUNG

§8 Abs. 1 lit. c PromO
Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO
Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO
Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO
Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, June 2021

Sounak Kar

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`.
Most of the graphics in this thesis are generated using `pgfplots` and `pgf/tikz`.
The bibliography is typeset using `biblatex`.