Hyper-Scalable Load Balancing: Novel Algorithms for Parallel-Server Systems

Mark van der Boor

This work was financially supported by The Netherlands Organization for Scientific Research (NWO) through the Gravitation Networks grant 024.002.003.

Hyper-Scalable Load Balancing: Novel Algorithms for Parallel-Server Systems

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-5228-3

Cover designed by Mark van der Boor

Printed by Gildeprint

Hyper-Scalable Load Balancing
Novel Algorithms for Parallel-Server Systems

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op vrijdag 26 maart 2021 om 16:00 uur

door

Mark van der Boor

geboren te Sliedrecht

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof. dr. J.J. Lukkien
1e promotor:	prof. dr. ir. S.C. Borst
2e promotor:	prof. dr. J.S.H. van Leeuwaarden
leden:	prof. dr. ir. I.J.B.F. Adan
	prof. dr. R.J. Boucherie (Universiteit Twente)
	prof. dr. M. Vlasiou
	dr. J. Anselmi (INRIA)
	prof. dr. T. Bonald (Telecom Paris)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Acknowledgments

The adventure started on January 6th, 2016, when I had a meeting planned with Sem and Johan to talk about the topic of my Master's project. Two days later I mentioned in an email that I probably came off as quite silent because there was a lot of information sent my way in the meeting, but I also mentioned that I was looking forward to working together.

Although I don't remember that much about this meeting, I can imagine how this meeting went, given the endless enthusiasm with which Sem and Johan supervised me not only during this project, but also during my PhD.

Sem, it was great to know that I could always count on you. I had the freedom to come up with my own ideas, and I am glad that you guided me in the right directions. I remember the long emails we shared when you worked in the US, and I'm afraid I will always remember our papers, fully decorated with comments in blue ink. Besides work, I'm grateful for the casual conversations we've had and for the house-party and dinner you organized. The trip to Cold Spring and Beacon together with Nancy will remain as a great memory, thank you both for this!

Johan, thank you as well! Somehow you exactly know how to present problems as challenges, and quite a few of our discoveries were created by my urge to solve challenges. The discussions we had were very encouraging.

I would also like to thank the members of my committee: Ivo Adan, Jonatha Anselmi, Thomas Bonald, Richard Boucherie, Maria Vlasiou, and Johan Lukkien for chairing my defense.

In 2018 I had the honor to go on an internship at IBM in the US. Mark and Soumyadip, thank you for the great supervision and the great experience!

I first visited my office, MF 4.073, on February 22th, 2016, where I was greeted by Angelos and Mayank. The office had a nice atmosphere but was quite silent, even after Viktoria joined later that year. But as time passed by, the office became more *gezellig* every day! Everyone had their perks but I'm glad you all became doctors. Jokes aside, I enjoyed the times we spent together in the office, at the coffee machine, playing table tennis and during our trips to Utrecht and Jerusalem. Martijn, Neeladri, Peter: hopefully you will get to enjoy the office as much as I did and the best of luck!

Rik, you deserve a separate paragraph, although all the previous things also refer to you of course. We spent a lot of time together in the office, in Jerusalem, in Lunteren and in/to Utrecht. Even after all this time I don't think I've ever heard you say something silly or foolish (although you mentioned to me that you don't think that's true, *that* might be the first foolish thing I've heard you say). Thanks for your advice regarding a lot of stuff, and thanks for having weekly lunch meetings with me when we weren't working in Eindhoven, these meetings kept me sane!

Another separate paragraph for Marko: out of all the different things I did during my PhD, teaching was arguably my favorite task! I'm glad you asked me to help with stochastic simulation. We were a great team and I learned a lot, probably because you let me free to decide how I wanted to give my lectures. And the students seemed to enjoy our lectures too!

There are of course many more persons of which I'm grateful to refer to as colleagues. Chantal, Ellen and Petra, thanks for helping me get started and for all the help along the way. Collin, Liron and Patty, I enjoyed organizing the YEQT workshop together, it went great. Céline, Debankur and Martin, collaborating with you was a blast and I've learned a lot from you. The daily lunches in the Eurandom lounge offered a great moment for reflection, so thanks to everyone with whom I've eaten lunch simultaneously and in particular to Bart, Fiona, Joost, Kay, Murtuza, Onno and Youri. And to all my colleagues: keep being awesome!

Then there's Networks; great thanks to all the members for organizing and participating in great events. Special shout-outs to Jan-Pieter for the great conversations we had and to Jaap for the (over-)ambitious ideas we discussed and carried out. The Networks days and Networks weeks were very enjoyable, it's a shame there are no Networks months or Networks years...

Ferdi, I'm glad you were able to endure all of the Christmas songs I played already in November during our radio shows, it is great to fool around on Saturdays and I hope we will keep doing so for a long time. Henk, I truly enjoyed collecting star-shaped objects with you within a certain fungus-filled fictional kingdom, and hope you will flourish within the Networks community. Tristan, hopefully you haven't lost too much sleep due to the random thoughts and messages I've been sending in the middle of the night, thanks for driving in rocket-powered battle-cars with me and for the common interest in fruit-dinosaurs. Of course also the best of luck with your PhD.

Ik noemde dat mijn avontuur in 2016 begon, maar mijn avontuur begon natuurlijk meer dan 27 jaar geleden. Ik ben trots op de kleine maar gezellige familie waarvan ik deel uitmaak: Esther, Michel, Jesse, Liam, Tante en Eric, inclusief diegenen die we gaandeweg helaas verloren hebben.

Tijger, mijn favoriete huisdier, fijn dat je me gezelschap hield tijdens de late uren. Ten slotte: mam, pap, bedankt voor alles dat jullie voor me hebben gedaan en ik ben blij dat jullie trots op me zijn, ik ben minstens zo trots op jullie!

Mark
Sliedrecht, February 2021

"*Goodbye everyone, I'll remember you all in therapy.*" — *Plankton, 2002*

Contents

1 Introduction **1**
1.1 Brief taxonomy of load balancing algorithms 1
1.2 Scalability 5
1.3 Mathematical techniques 6
1.3.1 Queueing networks 7
1.3.2 Fluid limits 8
1.3.3 Stochastic simulation 10
1.4 Main contributions 11

2 Load balancing with multiple dispatchers **15**
2.1 Introduction 15
2.2 Model description and key results 16
2.3 Jackson network representation 21
2.3.1 Proof of Theorem 2.1 23
2.4 Fluid limit in the blocking scenario 29
2.4.1 Derivation of fluid-limit equations 30
2.4.2 Fixed-point analysis 34
2.4.3 Enhancements 35
2.5 Fluid limit in the queueing scenario 36
2.5.1 Fixed-point analysis 38
2.5.2 Enhancements 40
2.6 Conclusion 42
2.A Proof of Proposition 2.1 42
2.B Proof of Proposition 2.2 46

3 Hyper-scalable load balancing with scheduled updates 49
3.1 Introduction 49
3.2 Model description and key results 50
3.2.1 Large-system performance 51
3.2.2 Synchronize or not? 52
3.2.3 Notation and preliminaries 53
3.3 Synchronized updates 55
3.3.1 Fluid-limit dynamics 55
3.3.2 Performance in the fluid limit 60
3.4 Asynchronous updates 70
3.4.1 Fluid-limit dynamics 70
3.4.2 Fixed-point analysis 76
3.5 Conclusion 81
3.A Proofs of Section 3.3.2 82
3.A.1 Proof of Lemma 3.2 82
3.A.2 Proof of Lemma 3.8 85
3.B Derivation of the fixed point 88

4 Optimal hyper-scalable load balancing with a strict queue limit 93
4.1 Introduction 93
4.2 Model description and key results 94
4.3 Universal upper bound 96
4.4 The hyper-scalable scheme 100
4.5 Simulation experiments and optimality benchmarks 105
4.5.1 Baseline version of the hyper-scalable scheme 105
4.5.2 Non-idling variant 107
4.5.3 Work-conserving variant 107
4.5.4 Comparison with the AUJSQ$^{\text{det}}(\delta)$ scheme 108
4.5.5 Non-exponential service times 109
4.6 Extension aimed at minimizing queue lengths 110
4.7 Closed queueing network and further proofs 114
4.7.1 Proof of Proposition 4.4 116
4.8 Conclusion 120

5 Load balancing with negative acknowledgments 123
5.1 Introduction 123
5.2 Model description and key results 124
5.3 Performance of the JOQ algorithm 126
5.4 Hybrid system using the JOQ and JIQ algorithms 130

5.4.1 System design 131
5.4.2 Choosing the system parameters 132
5.5 Numerical simulations and discussions 134
5.5.1 The choice of τ 135
5.5.2 Virtual queues or not 136
5.5.3 Non-exponential job size distributions 137
5.6 Conclusion 138
5.A Optimality of equidistant updates 139

Summary **149**

About the author **151**

Chapter 1

Introduction

Load balancing algorithms (LBAs) play a critical role in distributing service requests or jobs (e.g. compute jobs, data base lookups, file transfers) among servers or distributed resources in parallel-processing systems. The analysis and design of LBAs has attracted strong attention in recent years, mainly spurred by crucial scalability challenges arising in cloud networks and data centers with massive numbers of servers.

1.1 Brief taxonomy of load balancing algorithms

In this section we discuss several common load balancing algorithms. We focus on the most basic load balancing scenario which consists of N identical parallel servers and a single dispatcher where jobs arrive according to a Poisson process of rate $\lambda(N) = \lambda N$. Arriving jobs cannot be queued at the dispatcher, and must immediately be forwarded to one of the servers. Jobs are assumed to have unit-mean exponentially distributed service requirements, and the service discipline at each server is oblivious to the actual service requirements, for example first-come-first-served (FCFS). This construction is commonly referred to as the *supermarket model*.

The celebrated Join-the-Shortest-Queue (JSQ) policy has several strong stochastic optimality properties in the supermarket model. In particular, the JSQ policy achieves the minimum mean overall delay among all non-anticipating policies that do not have any advance knowledge of the service requirements [EVW80; Win77]. Scaling results for the JSQ scheme showing its merits in asymptotic regimes where the load approaches one as N grows large may be found in [EG18; GW19]. In order to implement the JSQ policy however, the dispatcher requires instantaneous knowledge of all the queue lengths, which may involve a prohibitive communication burden when the number of servers N is large.

JSQ(d) policies. The poor scalability of the JSQ policy has motivated consideration of JSQ(d) policies, where an incoming job is assigned to a server with the shortest queue among $d \geq 2$ servers selected uniformly at random. This involves an exchange of d messages per job irrespective of the number of servers N. Results in [Mit01; VDK96] indicate that even sampling as few as $d = 2$ servers yields significant performance enhancements over purely random assignment ($d = 1$) as N grows large, which is commonly referred to as the *power-of-two* or *power-of-choice* effect. These results also extend to heterogeneous servers, non-Markovian service requirements and loss systems [BLP10; BLP12; MKMG15; XDLS15].

The diversity parameter d thus induces a fundamental trade-off between the amount of communication overhead and the delay performance. Specifically, a random assignment policy does not entail any communication burden, but the mean waiting time remains *constant* as N grows large for any fixed $\lambda > 0$. In contrast, a nominal implementation of the JSQ policy (without maintaining state information at the dispatcher) involves N messages per job, but the mean waiting time *vanishes* as N grows large for any fixed $\lambda < 1$.

Although JSQ(d) policies with $d \geq 2$ yield major performance improvements over purely random assignment while reducing the communication burden by a factor $\mathrm{O}(N)$ compared to the JSQ policy, the mean waiting time *does not vanish* in the limit. Specifically, when jobs arrive at rate λN, the queue length distribution at each individual server exhibits super-exponential decay for any fixed $\lambda < 1$ as N grows large. Hence, no fixed value of d will provide asymptotically optimal delay performance. This is evidenced by results in [GTZ16] indicating that in the absence of any memory at the dispatcher the communication overhead per job *must increase* with N in order for any scheme to achieve a zero mean waiting time in the limit. The only exception arises in case of batch arrivals when the value of d and the batch size grow large in a specific proportion, as can be deduced from the arguments in [YSK15].

Now suppose that the number of servers sampled becomes a function of N: $d(N)$. In order to achieve zero mean waiting time in the limit, it is sufficient that $d(N) \to \infty$ as $N \to \infty$, so $d(N)$ could for example grow logarithmically with

N [LY18; MBLW16a; WW20]. While only $d(N) \ll N$ servers are sampled, the amount of communication overhead in terms of $d(N)$ must still grow with N. This may be explained from the fact that a large number of servers need to be sampled for each incoming job to ensure that at least one of them is found idle with high probability.

Dispatcher-driven versus server-driven algorithms. JSQ(d) policies are *dispatcher-driven* algorithms, in the sense that the dispatcher takes the initiative to collect state information from the various servers. In the above description we implicitly assumed that the dispatcher gathers queue length information to assign arriving jobs, but does not store it.

The performance of dispatcher-driven policies can be further improved by attaching memory to the dispatcher [AD20], which allows JSQ(d) with fixed d schemes to be asymptotically optimal just like JSQ, for sufficiently small load. A further performance analysis of JSQ(d) with memory is conducted in [HV20]. Memory also seems necessary in systems with multiple dispatchers and we will touch upon this shortly.

In contrast, in *server-driven* algorithms the various servers take the initiative to provide state information to the dispatcher, which may then be stored in memory and used to assign arriving jobs in the future. This can greatly reduce the communication overhead, while providing excellent performance.

JIQ policy. A key example of a server-driven algorithm is the so-called Join-the-Idle-Queue (JIQ) scheme [BB08; Lu+11] which has gained huge popularity recently and can be implemented through a simple token-based mechanism. Specifically, idle servers send tokens to the dispatcher to advertise their availability. When a job arrives and the dispatcher has tokens available, it assigns the job to one of the corresponding servers (and the token is discarded). When no tokens are available at the time of a job arrival, the job may either be discarded or forwarded to a randomly selected server. Note that a server only issues a token when a job completion leaves its queue empty, thus generating at most one message per job.

Remarkably, the JIQ scheme has the ability of the full JSQ policy to drive the queueing delay to zero as $N \to \infty$, even for generally distributed service requirements [FS17; Sto15]. Thus, the use of memory allows the JIQ scheme to achieve asymptotically optimal delay performance with low communication overhead. In particular, ensuring that jobs are assigned to idle servers whenever available is sufficient to achieve asymptotic optimality, and using any additional queue length information yields no meaningful performance benefits on the fluid level (we refer to Section 1.3 for a further introduction of fluid limits). An analysis of the JIQ scheme on the diffusion scale can be found in [MBLW16b].

Somewhat similar to the JIQ scheme is the Persistent-Idle scheme, which routes jobs to the server that has last been idle [AKMOV20]. This is specifically of interest in systems with heterogeneous servers.

Multiple-dispatcher scenarios. So far we have focused on a basic scenario with a single dispatcher, but it is not uncommon for systems to have multiple dispatchers. While the presence of multiple dispatchers does not affect some LBAs (for example dispatcher-driven without memory, such as JSQ(d) policies), it does matter for the JIQ scheme which uses memory at the dispatcher.

Early papers that consider scenarios with the JIQ scheme and multiple dispatchers almost exclusively assume that the loads at the various dispatchers are strictly equal [Lu+11; Mit16; Sto17]. In these cases the fluid limit, for suitable initial states, is the same as for a single dispatcher, and in particular the fixed point is the same, hence, the JIQ scheme continues to achieve asymptotically optimal delay performance with minimal communication overhead. The results in [Sto17] in fact show that the JIQ scheme remains asymptotically optimal even when the servers are heterogeneous, while it is readily seen that JSQ(d) policies cannot even be maximally stable in that case for any fixed value of d. In Chapter 2 we will analyze a scenario with multiple dispatchers that have different loads.

In the above setup the number of dispatchers is assumed to remain fixed as the number of servers grows large. Further natural scenarios would be for the number of dispatchers to scale with the number of servers, or servers may issue their availability tokens to the dispatchers already before they are idle, which appears beneficial at very high load [Mit16].

Alternative options to deal with heterogeneous loads could be to have idle servers issue copies of their availability tokens to multiple dispatchers [Lu+11]. [VKO20; ZSW20] consider schemes in which every dispatcher keeps local estimates of the queue lengths. Also related are [BCM19; Com19], in which the token-based implementation of the JIQ scheme is used to handle compatibility constraints between dispatchers and servers.

The literature on load balancing algorithms has ballooned in recent years. We refer to [BBLM18b] for a more comprehensive survey discussing related job assignment mechanisms, further model extensions and alternative asymptotic regimes (e.g. heavy-traffic and non-degenerate slowdown scalings).

1.2 Scalability

As mentioned earlier, implementation overhead (e.g. communication or memory usage involved in obtaining or storing state information) has emerged as a key concern in the design of LBAs, due to the immense size of cloud networks and data centers [Gan+14; MSY12; Pat+13]. Indeed, the fundamental challenge in load balancing is to achieve scalability: providing good delay performance, while only requiring low implementation overhead in large-scale deployments. The seminal paper [GTZ16] approached this challenge by imposing the natural performance criterion that the probability of non-zero delay vanishes as the number of servers grows large. It was shown that this can only be achieved with constant communication overhead per job when sufficient memory is available at the dispatcher.

Hyper-scalable load balancing. While many schemes may achieve this vanishing wait with one message per job (for example JIQ, relying on server-initiated updates), even that one message may still be prohibitive. This typically plays a role when jobs do not involve big computational tasks, but small data packets which require little processing, e.g. in 'Internet of Things' cloud environments. In such situations the sheer message exchange in providing queue length information may be disproportionate to the actual amount of processing required.

In this manuscript we introduce and analyze novel scalable LBAs which strike an optimal trade-off between performance and communication overhead. Specifically, LBAs are called 'hyper-scalable' when they can be implemented with a communication overhead of one or fewer messages per job, and preferably many fewer. The maximum permissible communication overhead of these schemes is usually well below the minimum requirement for vanishing delay in a many-server regime.

There are several ways to implement this hyper-scalable notion in algorithms [ZSW20]. One might think of simply adapting JIQ or JSQ(d) by not sending some of the messages, in order to have less than one message per job. In this manuscript, we will explore a different method: communication between server and dispatcher will be (more) decoupled from the arrival process and from job completions. Instead, a timer governs the communication process. When the timer ticks, information will be shared between server(s) and dispatcher.

The above-described notion is used in novel hyper-scalable schemes which we introduce in Chapters 3–5. In Chapter 3, the timer is completely independent from the system state. In Chapter 4, the timer starts when a server reaches a specific state, but the duration of the timer is fixed. In Chapter 5, the lack of a message from the server when the timer ticks, is used to update the local memory of the dispatcher.

Comparison with other schemes. The communication overhead of the hyper-scalable schemes in Chapters 3–5 can be described in terms of the update frequency per server, say δ. When the arrival rate per server equals $\lambda < 1$, the number of messages per job equals δ/λ, or δN per time unit, which can be easily tuned by varying the value of δ.

The JSQ(d) scheme, when implemented in a dispatcher-driven manner, requires d message exchanges per job, which amounts to $\lambda d N$ messages per time unit. When servers actively update their queue lengths to the dispatcher in the JSQ scheme or their idleness in the server-driven JIQ scheme, one needs less communication. In this case, any departing job needs to trigger the server to send an update to the dispatcher. This server-driven implementation requires one message per job or λN per time unit. Note that when queue lengths are large, not even all departing jobs need to trigger the server to send an update for the JIQ scheme, which reduces the communication per job slightly. In conclusion, the tunable communication overhead of δ/λ per job of the hyper-scalable algorithms are comparable with server-driven JSQ, JIQ and JSQ(d) schemes.

1.3 Mathematical techniques

We briefly discuss the main mathematical techniques that will be used in this manuscript, by illustrating how these techniques are used when analyzing the JIQ scheme. While the analyses in further chapters are more involved, they are in fact still based on the fundamental concepts presented here.

We consider the load balancing scenario as described in Section 1.1. When a job arrives at the dispatcher, the job will be immediately forwarded to one of the idle servers, if there are any. We consider two scenarios, referred to as *blocking* and *queueing*, depending on whether jobs are discarded or forwarded to a randomly selected server in the absence of any idle servers.

Markov process and state description. Because both the inter-arrival times and the service times are independent and exponentially distributed and thus memoryless, the evolution of the system occupancy can be modeled by a Markov process. A straightforward state description keeps track of the number of jobs in queue for each of the N servers. However, we do not keep track of the queue length of every individual server, but only of the number (or fraction) of servers that have $i = 0, 1, \ldots$ jobs in queue. This is sufficient to retain a Markovian state description since all the decisions in the JIQ scheme only discriminate between servers based on their current state and not their identity.

The state of this Markov process is given by $Y = (Y_0, Y_1, \ldots)$, with $Y_i \geq 0$ representing the number of servers with i jobs in queue and $\sum_i Y_i = N$. There are three possible transitions:

Transition (a). When there are idle servers available, i.e. $Y_0 > 0$, the system transitions from state Y to Y' with $Y_0' = Y_0 - 1$ and $Y_1' = Y_1 + 1$ at rate λN because of job arrivals.

Transition (b). Additionally, when there are no idle servers available ($Y_0 = 0$) in the queueing scenario, the system transitions from state Y to $Y^{(i)}$ with $Y_i^{(i)} = Y_i - 1$ and $Y_{i+1}^{(i)} = Y_{i+1} + 1$ at rate λY_i for all $i \geq 1$ because of job arrivals.

Transition (c). Furthermore, the system transitions from state Y to $Y^{(i)}$ with $Y_i^{(i)} = Y_i - 1$ and $Y_{i-1}^{(i)} = Y_i + 1$ at rate Y_i for all $i \geq 1$ because of service completions.

1.3.1 Queueing networks

In this subsection we consider the blocking scenario, so that the state of the system may be described by the vector (Y_0, Y_1), in which Y_0 stands for the number of idle servers and Y_1 stands for the number of servers with one job in queue. The idle servers receive jobs at rate λN (Transition (a)), and every busy server processes the job in queue at rate one (Transition (c)). Note that no servers have two or more jobs in queue as jobs are discarded when no idle server are available (and we assume that this condition holds for the initial state too).

The above-described system dynamics can be represented in terms of a closed product-form queueing network with a fixed population of size N, in which the servers act as customers traversing various nodes. Specifically, the queueing network consists of two nodes: a single-server node and an infinite-server node, and each server is in one of the two nodes. Servers in the single-server node move to the other node one by one at rate λN (Transition (a)). All servers in the infinite-server node move to the single-server node at rate one (Transition (c)). Translated back to the load balancing viewpoint: when a server is at the single-server node, it is idle; a server at the infinite-server node is busy.

According to [BD11; Kel11], the equilibrium distribution of the closed network has the product-form

$$\pi(Y_0, Y_1) = G^{-1}\left(\frac{1}{\lambda N}\right)^{Y_0}\frac{1}{Y_1!}$$

with $G = \sum_{a=0}^{N}\left(\frac{1}{\lambda N}\right)^{a}\frac{1}{(N-a)!}$. This distribution may also be verified by checking that it is indeed a solution of the balance equations and that it sums to one.

Note that the JIQ scheme in the blocking scenario may in fact also be described in terms of a birth-death process, but the framework of closed queueing networks is more general and allows us to analyze more intricate systems. The main reason for these types of systems to have a product-form equilibrium distribution is that servers cannot be busy *and* receive jobs simultaneously. Extensions of JIQ fall into this category as will be seen in Chapter 2, but also LBAs in which servers idle when they are about to receive jobs are candidates for product-form equilibrium distributions, as will be shown in Chapter 4.

1.3.2 Fluid limits

We examine so-called fluid limits where the evolution of suitably scaled state variables is considered as $N \to \infty$, and converges to a deterministic process governed by a set of differential equations. The fluid limits provide insight in the network dynamics on a macroscopic scale, and yield asymptotically exact approximations for several performance measures of interest, in particular blocking probabilities and waiting times.

We analyze the number of servers in specific states as a function of time. The vector of state variables at time t is given by $Y^N(t) = (Y_0^N(t), Y_1^N(t), \ldots)$ as before, in which the N in superscript illustrates the dependency on the total number of servers. We consider the scaled quantities $y^N(t) = (y_0^N(t), y_1^N(t), \ldots)$ with $y_i^N(t) = Y_i^N(t)/N$, where $y_i^N(t)$ denotes the fraction of servers that have $i = 0, 1, \ldots$ jobs in queue; $y^N(t)$ is the fluid-scaled state of the N-th system. If the stochastic process $\{y^N(t)\}_{t\geq 0}$ converges to a limit $\{y(t)\}_{t\geq 0}$ in some appropriate sense as $N \to \infty$, then the latter is called a fluid limit. Fluid limits have a certain commonality with laws of large numbers, and quite often turn out to be deterministic processes, which can be described in terms of a set of differential equations.

Proving a fluid limit requires showing convergence of a sequence of stochastic processes, and typically involves fairly advanced probabilistic machinery. In Markovian settings, a common proof technique relies on martingale representations of the process $\{Y^N(t)\}_{t\geq 0}$ in terms of Poisson processes, see for instance [HK94; PTW07]. However, in many situations, fluid limits have a relatively simple and insightful form, and can be derived heuristically, which will be how fluid limits will be used in this manuscript.

We illustrate this heuristic derivation for the JIQ scheme in the queueing scenario. We define $\lambda_1(t)$ to be the aggregate rate at which jobs are forwarded to idle servers at time t (Transition (a)), which equals λ when $y_0(t) > 0$, and $\min\{\lambda, y_1(t)\}$ when $y_0(t) = 0$, since servers that become idle at rate $y_1(t)$ may immediately be used to forward jobs to, but the maximum rate is trivially bounded by the arrival rate λ. Consequently, $\lambda - \lambda_1(t)$ is the aggregate rate at which jobs are forwarded to randomly selected servers (Transition (b)). Every individual server receives jobs at this rate, which makes the total rate that servers with i jobs in queue receive jobs equal to $[\lambda - \lambda_1(t)]y_i(t)$. Finally, service completions make servers with i jobs transition to having $i-1$ jobs at rate $y_i(t)$ (Transition (c)).

We obtain

$$\begin{aligned}
\frac{\mathrm{d}y_0(t)}{\mathrm{d}t} &= -\lambda_1(t) + y_1(t),\\
\frac{\mathrm{d}y_1(t)}{\mathrm{d}t} &= \lambda_1(t) - [\lambda - \lambda_1(t)]y_1(t) + y_2(t) - y_1(t),\\
\frac{\mathrm{d}y_i(t)}{\mathrm{d}t} &= [\lambda - \lambda_1(t)]y_{i-1}(t) - [\lambda - \lambda_1(t)]y_i(t) + y_{i+1}(t) - y_i(t), \quad i \geq 2,
\end{aligned}$$

with $\lambda_1(t) = \lambda - \max\{\lambda - y_1(t), 0\}\mathbb{1}\{y_0(t) = 0\}$ whenever y is differentiable at t.

By setting the derivatives to zero we obtain $y_0^* = 1-\lambda$, $y_1^* = \lambda$ and $y_i^* = 0$ for $i \geq 2$ as a fixed point assuming $\lambda \leq 1$. This fixed point corresponds to the scenario in which a fraction λ of the servers is busy with one job and a fraction of $1-\lambda$ is idle.

In this manuscript, we are typically interested in the question when $y_i^* = 0$ for all $i \geq 2$. We will demonstrate that this property holds in several cases and we will refer to it as ‘zero queueing in the limit’ or simply as vanishing queueing, because in this scenario a fraction one of all queues has zero or one jobs, so that the number of jobs waiting in queue is negligible on fluid scale. We will also refer to this scenario as ‘vanishing queueing delay’ or simply vanishing wait, by viewing the fixed point as a stationary distribution for a scaled system.

However, the translation between these two would involve showing that the many-server limit ($N \to \infty$) and stationary limit ($t \to \infty$) can be interchanged. While this is generally the case, it is typically difficult to prove, as it would require showing global asymptotic stability of the fluid limit using ad-hoc arguments, and for this reason this will not be explicitly pursued in this manuscript. Note that for statements about the average queue length, one would also need convergence in expectation. We refer to related work that outlines some of the necessary arguments [BBLM18a; Cec18; Muk18]. We use extensive simulation experiments however to illustrate that the results do in fact seem to hold in the stationary limit, even in systems with a moderately large number of servers.

Fluid limits will be used in Chapters 2 and 3 to investigate the behavior of the different schemes in many-server regimes.

1.3.3 Stochastic simulation

As alluded above, discrete-event simulations are used in all upcoming chapters to validate and illustrate results.

During the simulation for the JIQ scheme in the queueing scenario, a list is maintained that contains all idle servers. The simulation revolves around two types of 'events', the arrival events and the departure events. All future events are kept track of in the so-called future event set, in chronological order. In every step of the simulation, the first event in the future event set is selected and the implications of this event are applied to the system. The arrival event has only one property, the time at which it takes place. The departure event has two properties; the time at which it takes place and the job that will depart at this time.

When an arrival event is selected, a new job is created and a random server is selected (and removed) from the list of idle servers if non-empty, the job is added into the queue of the server and a departure event is scheduled for this specific job. If the list of idle servers is empty, a random server will be selected and the job is added to the queue of this server. The time at which this event takes place equals the current time plus a random sample from the service time distribution. Afterwards, an additional arrival event is scheduled and the time of this event equals the current time plus a random sample from the inter-arrival time distribution, in order to mimic the Poisson arrival process. When a departure event is selected, the corresponding job is removed from the queue and the corresponding server is added to the list of idle servers if the queue is empty or a departure event is scheduled for the first job in queue if the queue is non-empty.

The systems that we analyze in this manuscript are much more complicated than the JIQ scheme, and they often lead to more actions that need to be executed when a new event is selected, or even to the introduction of more event types. In hyper-scalable algorithms, a new update event may be added, which corresponds to a server sending its queue length to the dispatcher.

The simulations are programmed in Java in an object-oriented manner, and the results of the simulations are processed and visualized in Wolfram Mathematica.

1.4 Main contributions

We will now summarize the main contributions of this manuscript. We first discuss an extension of the JIQ scheme for multiple dispatcher, after which we turn to novel algorithms for the single-dispatcher case.

Load balancing with multiple dispatchers

In Chapter 2, we leverage product-form representations and fluid limits to establish that the blocking and wait no longer vanish when dispatchers have different loads, even for an arbitrarily low overall load. Remarkably, it is the least-loaded dispatcher that throttles tokens and leaves idle servers stranded, thus acting as bottleneck. We introduce two enhancements of the ordinary JIQ scheme where tokens are either distributed non-uniformly or occasionally exchanged among the various dispatchers. We prove that these extensions can achieve zero blocking and wait in the many-server limit, for any subcritical overall load and arbitrarily skewed load profiles. Extensive simulation experiments demonstrate that the asymptotic results are highly accurate, even for moderately sized systems.

This chapter is based on [BBL17].

Hyper-scalable load balancing with scheduled updates

In Chapter 3 we introduce and analyze a novel class of load balancing schemes where the various servers provide occasional queue updates to guide the load assignment. We show in Chapter 3 that the proposed schemes outperform JSQ(d) strategies with comparable communication overhead per job. The proposed schemes are particularly geared towards the sparse feedback regime with less than one message per job, where they outperform corresponding sparsified JIQ versions.

We investigate fluid limits for synchronous updates as well as asynchronous exponential update intervals. The fixed point of the fluid limit is identified in the latter case, and used to derive the queue length distribution. We also demonstrate that in the ultra-low feedback regime the mean stationary waiting time tends to a constant in the synchronous case, but grows without bound in the asynchronous case.

This chapter is based on [BBL19].

Optimal hyper-scalable load balancing with a strict queue limit

As discussed earlier, the trade-off between delay performance and implementation overhead has primarily been studied so far from the angle of the amount of overhead required to achieve asymptotically optimal performance, particularly vanishing delay in large-scale systems. In contrast, in Chapter 4, we focus on an arbitrarily sparse communication budget, possibly well below the minimum requirement for vanishing delay. Furthermore, jobs may only be admitted when a specific limit on the queue position of the job can be guaranteed.

The centerpiece of our analysis is a universal upper bound for the achievable throughput of any dispatcher-driven algorithm for a given communication budget and queue limit. We also propose a specific hyper-scalable scheme which can operate at any given message rate and enforce any given queue limit, while allowing the server states to be captured via a closed product-form network, in which servers act as customers traversing various nodes. The product-form distribution is leveraged to prove that the bound is tight and that the proposed hyper-scalable scheme is throughput-optimal in a many-server regime given the communication and queue limit constraints.

This chapter is based on [BBL20].

Load balancing with negative acknowledgments

Observe that in order for the waiting probability to vanish, the dispatcher must have (near-)certainty that a server is idle when assigning a job to that server. Thus, the dispatcher must have some kind of idleness certificate. Unless the dispatcher has access to the job sizes, as in [Ans19], or if the service requirements have bounded support, this suggests that one message per job is not only asymptotically sufficient, but also necessary in order for the queueing delay to vanish.

In Chapter 5 we introduce a scheme that allows for vanishing wait, while actually using strictly less than one message per job on average. This scheme exploits the crucial insight that an idle state need not be explicitly signaled by using a message, but can also be implicitly inferred by the dispatcher when *not receiving a message* from a server at a pre-arranged time instant. This paradigm is somewhat similar in spirit to negative acknowledgments in end-to-end transport protocols, but to the best of our knowledge has not been adopted in a load balancing context so far.

This chapter is based on [BZB20].

Schematic overview of the thesis

We conclude the introduction with a schematic overview of the properties of the proposed schemes and the techniques used to analyze them.

	Chapter 2; Multiple dispatchers	Chapter 3; Hyper-sc. load bal.	Chapter 4; Opt. h.-s. load bal.	Chapter 5; Negative acks.
Dispatcher-driven		X	X	
Server-driven	X	X		X
Multiple dispatchers	X			
Hyper-scalable		X	X	X
Queueing networks	X		X	
Fluid limits	X	X		
Stochastic coupling	X			X
Stochastic sim.	X	X	X	X

Chapter 2

Load balancing with multiple dispatchers

Based on:

[BBL17] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. “Load balancing in large-scale systems with multiple dispatchers”. In: *Proc. INFOCOM ’17*. 2017

2.1 Introduction

In this chapter we examine the performance of the JIQ scheme (see Section 1.1) in systems with multiple dispatchers with heterogeneous loads. We distinguish two scenarios, referred to as *blocking* and *queueing*, depending on whether jobs are discarded or forwarded to a randomly selected server in the absence of any tokens at the dispatcher.

We use queueing networks with exact product-form distributions (see Section 1.3.1) and fluid-limit techniques (see Section 1.3.2) to establish that the blocking and wait no longer vanish for asymmetric dispatcher loads as the total number of servers grows large. In fact, even for an arbitrarily small degree of skewness and low overall load, the blocking and wait are strictly positive in the limit. This contrasts with the observation that the JIQ scheme remains asymptotically optimal in scenarios with multiple homogeneous dispatchers even when the servers are heterogeneous, as mentioned in Section 1.1. We show that, surprisingly, it is the least-loaded dispatcher that acts as a bottleneck and throttles the flow of tokens. The accumulation of tokens at the least-loaded dispatcher hampers the visibility of idle servers to the heavier-loaded dispatchers, and leaves idle servers stranded while jobs queue up at other servers.

In order to counter the above-described performance degradation for asymmetric dispatcher loads, we introduce two extensions to the basic JIQ scheme. In the first mechanism tokens are not uniformly distributed among dispatchers, but for example in proportion to the respective loads. We prove that this enhancement achieves zero blocking and wait in a many-server regime, for any subcritical overall load and arbitrarily skewed load patterns. In the second approach, tokens are continuously exchanged among the various dispatchers at some exponential rate. We establish that for any load profile with subcritical overall load there exists a finite token exchange rate for which the blocking and wait vanish in the many-server limit. Extensive simulation experiments (see Section 1.3.3) are conducted to corroborate these results, indicating that they apply even in moderately sized systems.

Organization of the chapter. The remainder of this chapter is organized as follows. In Section 2.2 we present a detailed model description, specify the two proposed enhancements and state the main results. In Section 2.3 we describe how the blocking scenario can be represented in terms of a closed Jackson network, and leverage the associated product-form distribution to obtain an insightful formula for the blocking probability. We then turn to a fluid-limit approach in Section 2.4 to analyze the two proposed enhancements in the blocking scenario. A similar analysis is adopted in Section 2.5 in the queueing scenario to obtain results for the basic model and the enhanced variants. Finally, in Section 2.6 we make some concluding remarks and briefly discuss future research directions.

2.2 Model description and key results

We consider a system with N parallel identical servers and a fixed set of R (not depending on N) dispatchers, as depicted in Figure 2.1. Jobs arrive at dispatcher r as a Poisson process of rate $\alpha_r \lambda N$, with $\alpha_r > 0$, $r = 1,\ldots,R$, $\sum_{r=1}^{R} \alpha_r = 1$, and λ denoting the job arrival rate per server. For conciseness, we denote $\alpha = (\alpha_1,\ldots,\alpha_R)$, and without loss of generality we assume that the dispatchers are indexed such that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_R$. The job processing requirements are independent and exponentially distributed with unit mean at each of the servers.

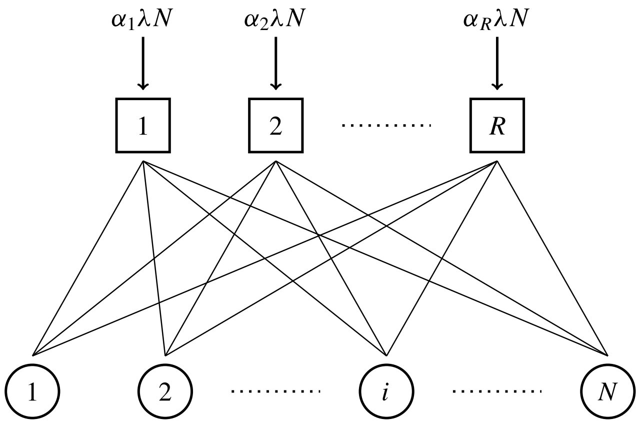


Figure 2.1: Schematic view of the model with R dispatchers and N servers.

When a server becomes idle, it sends a token to one of the dispatchers selected uniformly at random, advertising its availability. When a job arrives at a dispatcher which has tokens available, one of the tokens is selected, and the job is immediately forwarded to the corresponding server.

We distinguish two scenarios referred to as the *blocking* and *queueing* scenario, respectively, which differ in case a job arrives at a dispatcher without available tokens. In the blocking scenario, the incoming job is blocked and instantly discarded. In the queueing scenario, the arriving job is forwarded to one of the servers selected uniformly at random. If the selected server happens to be idle, then the outstanding token at one of the other dispatchers is revoked.

In the queueing scenario we assume $\lambda < 1$, which is not only necessary but also sufficient for stability. It is not difficult to show that the joint queue length process is stochastically majorized by a case where each job is sent to a uniformly at random selected server. In the latter case, the system decomposes into N independent M/M/1 queues, each of which has load $\lambda < 1$ and is stable.

We will use a different notation than we introduced in Section 1.3.2, since we will have to track at which dispatcher the token of a server is present. Denote by $X_0(t)$ the number of busy servers and by $X_r(t)$ the number of tokens held by dispatcher r at time t, $r = 1,\ldots,R$. Note that $\sum_{r=0}^{R} X_r(t) = N$ for all t, because either a server is busy or it has an outstanding token at one of the dispatchers. Also, denote by $Y_i(t)$ the number of servers with i jobs (including a possible job being processed) at time t, $i \geq 0$, so that $X_0(t) \equiv \sum_{i=1}^{\infty} Y_i(t)$.

In the blocking scenario, no server can have more than one job, i.e., $Y_i(t) = 0$ for all $i \geq 2$ and $X_0(t) = Y_1(t)$. Because of the symmetry among the servers, the state of the system can be described by the vector $X(t) = (X_0(t), X_1(t), \ldots, X_R(t))$, and $\{X(t)\}_{t\geq 0}$ evolves as a Markov process, with state space $S := \{n \in \mathbb{N}^{R+1} : \sum_{i=0}^{R} n_i = N\}$. Likewise, in the queueing scenario, the state of the system can be described by the vector $U(t) = (Y(t), X_1(t), \ldots, X_R(t))$ with $Y(t) = (Y_i(t))_{i\geq 0}$, and $\{U(t)\}_{t\geq 0}$ also evolves as a Markov process.

Denote by $B(R, N, \lambda, \alpha)$ the steady-state blocking probability of an arbitrary job in the blocking scenario. Also, denote by $W(R, N, \lambda, \alpha)$ a random variable with the steady-state waiting-time distribution of an arbitrary job in the queueing scenario, for which we will use the fixed point of the fluid limit without proof (see Section 1.3.2). We include R, N, λ and α in the notation in order to reflect the dependence of both performance metrics on these system parameters. However, we will occasionally suppress (some of) the arguments for brevity when these are fixed or otherwise clear from the context.

In Section 2.3 we will prove the following theorem for the blocking scenario.

Theorem 2.1 (Least-loaded dispatcher determines blocking). *As $N \to \infty$,*

$$B(R, N, \lambda, \alpha) \to \max\{1 - R\alpha_R, 1 - 1/\lambda\}.$$

Theorem 2.1 shows that in the many-server limit the system performance in terms of blocking is either determined by the relative load of the least-loaded dispatcher, or by the aggregate load. This may be informally explained as follows. Let $\bar{x}_0$ be the expected fraction of busy servers in steady state, so that each dispatcher receives tokens on average at a rate $\bar{x}_0 N/R$. We distinguish two cases, depending on whether a positive fraction of the tokens reside at the least-loaded dispatcher R in the limit or not. If there is a positive fraction, then the job arrival rate $\alpha_R \lambda N$ at dispatcher R must equal the rate $\bar{x}_0 N/R$ at which it receives tokens, i.e., $\bar{x}_0/R = \alpha_R \lambda$. Otherwise, the job arrival rate $\alpha_R \lambda N$ at dispatcher R must be no less than the rate $\bar{x}_0 N/R$ at which it receives tokens, i.e., $\bar{x}_0/R \leq \alpha_R \lambda$. Since dispatcher R is the least-loaded, it then follows that $\bar{x}_0/R \leq \alpha_r \lambda$ for all $r = 1, \ldots, R$, which means that the job arrival rate at all the dispatchers is higher than the rate at which tokens are received. Thus the fraction of tokens at each dispatcher is zero in the limit, i.e., the fraction of idle servers is zero, implying $\bar{x}_0 = 1$. Combining the two cases, and observing that $\bar{x}_0 \leq 1$, we conclude that $\bar{x}_0 = \min\{R\alpha_R \lambda, 1\}$. Because of Little's law, $\bar{x}_0$ is related to the

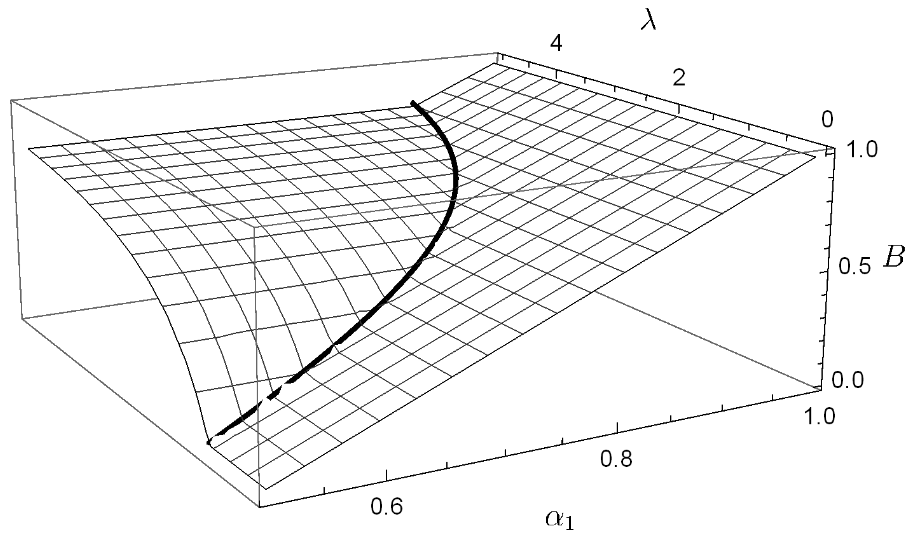


Figure 2.2: Blocking probability $B(2, N, \lambda, (\alpha_1, 1-\alpha_1))$ obtained by (2.2), for $R = 2$ dispatchers and $N = 10^5$ servers as function of λ and α_1.

blocking probability B as $\bar{x}_0 = \lambda(1-B)$. This yields $1-B = \min\{R\alpha_R\lambda, 1/\lambda\}$, or equivalently, $B = \max\{1-R\alpha_R, 1-1/\lambda\}$ as stated in Theorem Theorem 2.1.

The above explanation also reveals that in the limit dispatcher R (or the set of least-loaded dispatchers in case of ties) inevitably ends up with all the available tokens, if any. The accumulation of tokens hampers the visibility of idle servers to the heavier-loaded dispatchers, and leaves idle servers stranded while jobs queue up at other servers.

Figure 2.2 illustrates Theorem 2.1 for $R = 2$ dispatchers and $N = 10^5$ servers, and clearly reflects the two separate regions in which the blocking probability depends on either α_R or λ. The line represents the cross-over curve $R\alpha_R = 2\alpha_2 = 2(1-\alpha_1) = 1/\lambda$.

In Section 2.5 we will establish the following theorem for the queueing scenario.

Theorem 2.2 (Mean waiting time). *For $\lambda < 1$ and $N \to \infty$,*

$$\mathbb{E}[W(R,N,\lambda,\alpha)] \to \frac{\lambda_2(R,\lambda,\alpha)}{1-\lambda_2(R,\lambda,\alpha)},$$

where

$$\lambda_2(R,\lambda,\alpha) = 1 - \frac{1-\lambda\sum_{i=1}^{r^*}\alpha_i}{1-\lambda r^*/R}$$

with

$$r^* = \sup\{r|\alpha_r > \frac{1}{R}\frac{1-\lambda\sum_{i=1}^{r}\alpha_i}{1-\lambda r/R}\}$$

and the convention that $r^* = 0$ *if* $\alpha_1 = \ldots = \alpha_R = 1/R$.

λ_2 can be interpreted as the rate at which jobs are forwarded to randomly selected servers. Furthermore, dispatchers $1,\ldots,r^*$ receive tokens at a lower rate than the incoming jobs, and in particular $\lambda_2^* = 0$ if and only if $r^* = 0$. When $R = 2$, Theorem 2.2 simplifies to

$$\mathbb{E}[W(2,N,\lambda,(1-\alpha_2,\alpha_2))] \to \frac{\lambda(1-2\alpha_2)}{2-2\lambda(1-\alpha_2)}.$$

When the arrival rates at all dispatchers are strictly equal, i.e., $\alpha_r = 1/R$ for all $r = 1,\ldots,R$, Theorems 2.1 and 2.2 indicate that the stationary blocking probability and the mean waiting time asymptotically vanish in a regime where the total number of servers N grows large, which is in agreement with the results in [Sto17]. However, when the arrival rates at the various dispatchers are not perfectly equal, so that $\alpha_R < 1/R$, the blocking probability and mean wait are strictly positive in the limit, even for arbitrarily low overall load and an arbitrarily small degree of skewness in the arrival rates. Thus, the basic JIQ scheme fails to achieve asymptotically optimal performance when the dispatcher loads are not strictly equal.

In order to counter the above-described performance degradation for asymmetric dispatcher loads, we propose two enhancements.

Enhancement 2.1 (Non-uniform token allotment). *When a server becomes idle, it sends a token to dispatcher r with probability* β_r.

Enhancement 2.2 (Token exchange mechanism). *Any token is transferred to a uniformly randomly selected dispatcher at rate* ν.

Note that the token exchange mechanism only creates a constant communication overhead per job as long as the rate ν does not depend on the number of servers N, and thus preserves the scalability of the basic JIQ scheme.

The above enhancements can achieve asymptotically optimal performance for suitable values of the β_r parameters and the exchange rate ν, as stated in the next proposition.

Theorem 2.3 (Vanishing blocking and waiting). *For any $\lambda < 1$, the stationary blocking probability in the blocking scenario and the mean waiting time in the queueing scenario asymptotically vanish as $N \to \infty$, upon using Enhancement 2.1 with $\beta_r = \alpha_r$ for $r = 1, \ldots, R$ or Enhancement 2.2 with $\nu \geq \frac{\lambda}{1-\lambda}(\alpha_1 R - 1)$.*

The minimum value of ν required in the blocking scenario may be intuitively understood as follows. Zero blocking means that a fraction λ of the servers must be busy, and thus a fraction $1 - \lambda$ of the tokens reside with the various dispatchers, while the heaviest loaded dispatcher 1 receives enough tokens for all incoming jobs: $\alpha_1 \lambda \leq \lambda / R + \nu(1-\lambda)/R$ which is satisfied by the given minimum value of ν.

A similar reasoning applies to the queueing scenario, although in that case the number of servers with exactly one job no longer equals the number of busy servers, and a different approach is needed.

In order to establish Theorems 2.2 and 2.3, we examine in Sections 2.4 and 2.5 the fluid limits for the blocking and queueing scenarios, respectively. Rigorous proofs to establish weak convergence to the fluid limit are omitted, but can be constructed along similar lines as in [HK94; PTW07]. Simulation experiments will be conducted to verify the accuracy of the fluid-limit approximations, and show an excellent match, even in systems with a small number of servers.

2.3 Jackson network representation

In this section we describe how the blocking scenario can be represented in terms of a closed Jackson network. We leverage the associated product-form distribution to express the asymptotic blocking probability as a function of the aggregate load and the minimum load across all dispatchers, proving Theorem 2.1.

We view the system dynamics in the blocking scenario in terms of the process $\{X(t)\}_{t \geq 0}$ as a fixed total population of N tokens that circulate through a network of $R + 1$ stations. Specifically, the tokens can reside either at station 0, meaning that the corresponding server is busy, or at some station $r = 1,\dots,R$, indicating that the corresponding server is idle and has an outstanding token with dispatcher r.

Let $s_i(k)$ denote the service rate at station i when there are k tokens present. Then $s_0(k) = k$ and $s_r(k) = 1$ for $r = 1,\dots,R$. The service times are exponentially distributed at all stations, but station 0 is an infinite-server node with mean service time $\mu_0^{-1} = 1$, while station r is a single-server node with mean service time $\mu_r^{-1} = (\alpha_r \lambda N)^{-1}$, $r = 1,\dots,R$. The routing probabilities p_{ij} of tokens moving from station i to station j are given by $p_{r0} = 1$ for $r = 1,\dots,R$ and $p_{0r} = 1/R$ for $r = 1,\dots,R$. With γ_i denoting the throughput of tokens at station i, the traffic equations

$$\gamma_i = \sum_{j=0}^{R} \gamma_j p_{ji}, \quad i = 0,\dots,R$$

uniquely determine the relative values of the throughputs, i.e., the throughputs up to a common scaling factor.

Let $\pi(n) := \lim_{t\to\infty} \mathbb{P}\{X(t) = n\}$ be the stationary probability that the process $\{X(t)\}_{t\geq 0}$ resides in state $n \in S$. The theory of closed Jackson networks [Kel11] implies

$$\pi(n_0, n_1, \dots, n_R) = G^{-1} \prod_{i=0}^{R} \frac{(\gamma_i/\mu_i)^{n_i}}{\prod_{m=1}^{n_i} s_i(m)}, \tag{2.1}$$

with G a normalization constant.

The blocking probability can then be expressed by summing the probabilities $\pi(n)$ over all the states with $n_r = 0$ where no tokens are available at dispatcher r, and weighting these with the fractions α_r, $r = 1,\dots,R$:

$$B(R, N, \lambda, \alpha) = \sum_{r=1}^{R} \alpha_r \sum_{n \in \{n \mid n_r = 0\}} \pi(n). \tag{2.2}$$

Despite this rather complicated expression, Theorem 2.1 provides a compact characterization of the blocking probability in the many-server limit $N \to \infty$, as will be proved in the following parts.

Remark. While we assumed exponentially distributed service times, the infinite-server node is symmetric and thus the product-form solution in (2.2) as well as Theorem 2.1 still hold for phase-type distributions [Kel11].

2.3.1 Proof of Theorem 2.1

The proof of Theorem 2.1 uses stochastic coupling, for which we define a 'better' system S^+ and a 'worse' system S^-. Both systems are amenable to analysis and have an identical blocking probability in the many-server limit $N \to \infty$, as will be stated in propositions. Via coupling it follows that $B_{S^+} \leq B_S \leq B_{S^-}$ and $\lim_{N\to\infty} B_{S^+} = \lim_{N\to\infty} B_{S^-} = \max\{1 - R\alpha_R, 1 - 1/\lambda\}$, so that

$$\lim_{N\to\infty} B(R, N, \lambda, \alpha) = \max\{1 - R\alpha_R, 1 - 1/\lambda\}.$$

First we will analyze the 'better' and the 'worse' system. Afterwards, we will focus on the coupling arguments.

Inhomogeneous blocking scenario with two dispatchers

The better system merges the first $R-1$ dispatchers into one superdispatcher, which results in two dispatchers with arrival rates $(1-\alpha_R)\lambda N$ and $\alpha_R\lambda N$, respectively. However, in contrast to the original blocking scenario, when a job is completed and leaves a server idle, a token is not sent to either dispatcher with equal probability. Instead, tokens are sent to the superdispatcher with probability $\frac{R-1}{R}$.

First a description of the blocking probability will be deduced. Consider the blocking scenario with two dispatchers and N servers. Say that a token is sent to dispatcher i with probability β_i. Without loss of generality, assume $\frac{\alpha_1}{2\beta_1} \geq \frac{\alpha_2}{2\beta_2}$. Since we obtain a closed Jackson network (see (2.1)), we have the stationary distribution

$$\pi(n_0, n_1, n_2) = G^{-1} \frac{(\lambda N)^{n_1}}{(\frac{\alpha_1}{2\beta_1}\lambda N)^{n_1}} \frac{(\lambda N)^{n_2}}{(\frac{\alpha_2}{2\beta_2}\lambda N)^{n_2}} \frac{(2\lambda N)^{n_0}}{n_0!} \tag{2.3}$$

in which G is the normalization constant.
With $\gamma_1 = \frac{\alpha_1}{2\beta_1}$ and $\gamma_2 = \frac{\alpha_2}{2\beta_2}$, we write (2.3) as

$$\pi(n_0, n_1, n_2) = G^{-1} \frac{(2\lambda N)^{n_0}}{\gamma_1^{n_1}\gamma_2^{n_2} n_0!} \tag{2.4}$$

with

$$G = \sum_{n_0=0}^{N} \sum_{n_1=0}^{N-n_0} \frac{(2\gamma_2\lambda N)^{n_0}}{\gamma_2^N n_0!} \left(\frac{\gamma_2}{\gamma_1}\right)^{n_1}. \tag{2.5}$$

We assume that $\gamma_1 \neq \gamma_2$, so that $\gamma_2/\gamma_1 \neq 1$. When $\gamma_1 = \gamma_2$, the blocking probability does not depend on α_i or β_i. In that case, we obtain a symmetric system for $R = 2$ and refer to the analysis of the symmetric blocking scenario.

We rewrite (2.5) as

$$\begin{aligned} G &= \sum_{n=0}^{N} \frac{(2\gamma_2 \lambda N)^n}{\gamma_2^N n!} \frac{1-\left(\frac{\gamma_2}{\gamma_1}\right)^{N-n+1}}{1-\frac{\gamma_2}{\gamma_1}} \\ &= \frac{1}{\gamma_2^N} \frac{1}{\gamma_1 - \gamma_2} \left[\gamma_1 \sum_{n=0}^{N} \frac{(2\gamma_2 \lambda N)^n}{n!} - \gamma_2 \left(\frac{\gamma_2}{\gamma_1}\right)^N \frac{(2\gamma_2\lambda N)^n}{n!} \right]. \end{aligned} \tag{2.6}$$

Using (2.4) and (2.6) in (2.2) gives the blocking probability

$$\begin{aligned} B(2, N, \lambda, \alpha_1, \alpha_2, \beta_1, \beta_2) &= \alpha_1 \sum_{n=0}^{N} \pi(n, 0, N-n) + \alpha_2 \sum_{n=0}^{N} \pi(n, N-n, 0) \\ &= G^{-1} \left[\alpha_1 \sum_{n=0}^{N} \frac{\left(2\gamma_2\lambda N\right)^n}{\gamma_2^N n!} + \alpha_2 \sum_{n=0}^{N} \frac{\left(2\gamma_1 \lambda N\right)^n}{\gamma_1^N n!} \right] \\ &= (\gamma_1 - \gamma_2) \left[\frac{\alpha_1 \sum_{n=0}^{N} \frac{(2\gamma_2\lambda N)^n}{n!} + \alpha_2 \left(\frac{\gamma_2}{\gamma_1}\right)^N \sum_{n=0}^{N} \frac{(2\gamma_1\lambda N)^n}{n!}}{\gamma_1 \sum_{n=0}^{N} \frac{(2\gamma_2\lambda N)^n}{n!} - \gamma_2 \left(\frac{\gamma_2}{\gamma_1}\right)^N \sum_{n=0}^{N} \frac{(2\gamma_1\lambda N)^n}{n!}} \right]. \end{aligned} \tag{2.7}$$

We establish the following result for the blocking scenario with $R = 2$ enhanced with non-uniform token allotment.

Proposition 2.1 (Limiting blocking probability for $R = 2$). *Consider a system with 2 dispatchers, dispatcher r receives a fraction α_r of the jobs, and a token is sent to dispatcher r with probability β_r. The following expression holds for the blocking probability for large systems.*

$$\lim_{N\to\infty} B(2, N, \lambda, \alpha_1, \alpha_2, \beta_1, \beta_2) = \max\left\{ \beta_1 \left(\frac{\alpha_1}{\beta_1} - \frac{\alpha_2}{\beta_2} \right), 1 - 1/\lambda \right\}.$$

We visualize Proposition 2.1 in Figure 2.3. We see that the blocking probability increases with λ. Furthermore, the smallest blocking probability is obtained for $\alpha_1 = \beta_1$.

We apply Proposition 2.1 with $\alpha_1 = (1-\alpha_R)$, $\alpha_2 = \alpha_R$, $\beta_1 = \frac{R-1}{R}$ and $\beta_2 = 1/R$ to obtain a blocking probability equal to the probability in Theorem 2.1.

The proof of Proposition 2.1 is given in Section 2.A.

Symmetric blocking scenario with R dispatchers

The worse system thins the incoming rates of jobs at the dispatchers, so that some jobs are blocked, irrespective of whether or not the dispatcher has any tokens available. This thinning process is defined as follows: a job arriving at dispatcher r is blocked with probability

$$\frac{\alpha_r - \alpha_R}{\alpha_r} + \frac{\max\{0, \alpha_R - 1/(R\lambda)\}}{\alpha_r}.$$

This thinning process is designed in such a way that the system with admitted jobs behaves as a system with total arrival rate $\lambda \le 1$ in which all arrival rates are equal ($\alpha_1 = \ldots = \alpha_R$), for which the blocking probability can be computed explicitly.

The stationary distribution of the system is given by

$$\pi(n_0, n_1, \ldots, n_R) = G_1^{-1} \frac{(\lambda N)^{n_1}}{(\alpha_1 \lambda N)^{n_1}} \cdots \frac{(\lambda N)^{n_R}}{(\alpha_R \lambda N)^{n_R}} \frac{(R\lambda N)^{n_0}}{n_0!} = G_2^{-1} \frac{(\lambda N)^{n_0}}{n_0!}$$

in which G_1 and G_2 are normalization constants. Notice that since $n_1 + \ldots + n_R = N - n_0$, π does not depend on the locations of individual tokens in terms of the

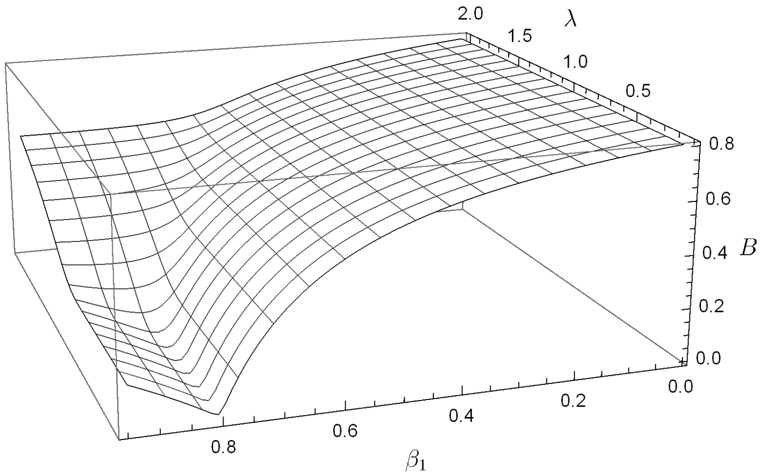


Figure 2.3: The blocking probability as in Proposition 2.1 for $\alpha_1 = 0.8$.

values of $n_1, \ldots, n_R$. For a given n_0, we have $N - n_0$ tokens that can be at any of the R dispatchers. These tokens can be distributed in $\binom{N-n_0+R-1}{R-1}$ ways, and hence

$$G = \sum_{n=0}^{N} \binom{N+R-1-n}{R-1} \frac{(\lambda N)^n}{n!}.$$

We compute the probability that dispatcher 1 has no tokens, which implies that an arriving job would be blocked. If we assume that n_0 servers are busy, then $N - n_0$ tokens are at dispatchers $2, 3, \ldots, R$. There are a total of $\binom{N+R-2-n}{R-2}$ states for which that is the case. Therefore, the probability that there are no tokens available at dispatcher 1 is

$$B(R,N,\lambda) = \frac{\sum_{n=0}^{N} \binom{N+R-2-n}{R-2} \frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N} \binom{N+R-1-n}{R-1} \frac{(\lambda N)^n}{n!}}. \tag{2.8}$$

Note that this derivation is also valid for $R = 1$, using the convention $\binom{-1}{-1} = 1$. The expression in (2.8) gives the blocking probability at one of the dispatchers. Since dispatchers are indistinguishable, (2.8) also is the blocking probability of the entire system.

Proposition 2.2 (Vanishing blocking probability in symmetric systems)**.** *Consider a system with R equal dispatchers. For $\lambda \leq 1$,*

$$\lim_{N\to\infty} B(R,N,\lambda) = 0.$$

The proof of Proposition 2.2 is given in Section 2.B.

Proposition 2.2 tells us that the system after thinning has a limiting blocking probability of 0, so that in the limit, jobs are only blocked due to the thinning itself. The total blocking probability is

$$\begin{aligned}\sum_r \alpha_r \left(\frac{\alpha_r - \alpha_R}{\alpha_r} + \frac{\max\{0, \alpha_R - \frac{1}{R\lambda}\}}{\alpha_r} \right) &= 1 - R\alpha_R + \max\{0, R\alpha_R - 1/\lambda\} \\ &= \max\{1 - R\alpha_R, 1 - 1/\lambda\},\end{aligned}$$

which is equal to the blocking probability in Theorem 2.1.

Stochastic coupling

With coupling, one can show that the blocking probability of the ‘better system’ is lower and of the ‘worse system’ is higher, which completes the proof. Specifically, when the arrival moments, the service times and the token-allotment are coupled, the number of tokens used at each dispatcher by time t is always lower in the worse system and higher in the better system. Intuitively, the tokens at dispatchers 1 to $R-1$ are consolidated in the better system. If there is at least one token amongst these dispatchers, any job arriving at any of the dispatchers can make use of a token. In the original system, a job is blocked when the token amongst the first $R-1$ dispatchers, is not present at the dispatchers at which a job arrives. The worse system performs obviously worse, since blocking jobs beforehand has no benefits for the acceptance of jobs.

We will now discuss the details of the coupling proof. Consider system S with N servers, R dispatchers and general arrival rates.

Better system. S^+ represents the system with only two dispatchers. We will add + in superscript to a variable/function, to indicate that this variable/function is a property of system S^+, instead of system S. We will assume that both systems have all N servers busy at time 0. We will couple the systems so that the ith arrival at dispatcher r occurs at the same time in both systems. Also, the ith service time in both systems is of equal length. Finally, after the jth service completion, a token is sent to the same dispatcher in both systems. When a token is sent to dispatcher $r \neq R$ in S, the token is sent to superdispatcher 1 in S^+.

We introduce the functions

- $T_r(t)$; the cumulative number of tokens that have been used at time t at dispatcher r,
- $N_r(n)$; the number of tokens that are sent to dispatcher r out of the first n tokens sent,
- $\xi_r(t)$; the number of tokens at dispatcher r at time t.

These functions are indexed by r, where r can indicate one of the dispatchers $1,\dots,R$, or a collection of dispatchers. If the index r refers to a collection of dispatchers, we simply sum all dispatchers in this collection. We introduce the collections $A = \{1,\dots,R-1\}$ and $X = \{1,\dots R\}$ and the following additional functions:

- $S(t)$; the number of service completions by time t,
- $\mu(n)$; the service time of the n'th job that starts its service,
- $t(n)$; the time at which a token is used for the n'th time,

- $s(n)$; the time at which a service is completed for the n'th time.

We define $T_r(0) = S(0) = 0$. We now claim that

$$(T_A(t), T_R(t)) \leq_{\text{st}} \left(T_1^+(t), T_R^+(t)\right) \quad \forall t \geq 0, \tag{2.9}$$

in which we define $X \leq_{\text{st}} Y$ as $\mathbb{P}\{X \leq k\} \geq \mathbb{P}\{Y \leq k\}$ for all k. Assuming an equal initial state, the claim is true for $t = 0$. Next, we use mathematical induction and assume that the claim is true for some $t > 0$. We will show that after any possible event (arrival or service completion), the inequality still holds. First, we establish the following relations:

(a) $\xi_A(t) = N_A(S(t)) - T_A(t)$,

(b) $\xi_R(t) = N_R(S(t)) - T_R(t)$,

(c) $t(n) = \inf\{t | T_X(t) \geq n\}$,

(d) $s(n) = \inf\left\{t \left| \sum_{i=1}^{\infty} 1\left\{t(i) + \mu(i) \leq t\right\} \geq n\right.\right\}$,

(e) $S(t) = \sum_{i=1}^{\infty} 1\{s(i) \leq t\}$.

Relations (a) and (b) shows that the number of tokens is the number of arrived tokens minus the number of tokens sent. Relation (c) shows that the time at which the nth token is used is the first time at which n tokens are used. Relation (d) requires more thought. First, $t_i + \mu_i$ is the time at which the job which used the ith token, has completed its service. Then, $\sum_{i=1}^{\infty} 1\left\{t(i) + \mu(i) \leq t\right\}$ counts the number of service completions before time t. The first time that this number reaches n, is $s(n)$. Finally, in Relation (e), we check for all jobs if they have finished service before time t, and count the number of those jobs to get the number of service completions by time t.

With these relations, we can show that (2.9) holds at all times. First, note that (2.9) implies that $t(n) \leq t^+(n)$ and $s(n) \leq s^+(n)$ for all n (Relations (c) and (d)), so that $S(t) \leq S^+(t)$ (Relation (e)) and $N_r(S(t)) \leq N_r(S^+(t))$, where r can indicate a single dispatcher or any collection of dispatchers. Trivially, when $T_A(t) < T_1^+(t)$, an arrival at any dispatcher $1, \ldots R-1$ will not influence the equation with $\leq$. Vice versa, after an arrival at dispatcher R when $T_R(t) < T_R^+(t)$, the equation will still hold with $\leq$. Also, a service completion will not change the number of tokens used. So, we have to consider two possible events:

E1 Arrival at dispatcher $1, \ldots, R-1$ when $T_A(t) = T_1^+(t)$ and $T_R(t) \leq T_R^+(t)$.

E2 Arrival at dispatcher R when $T_A(t) \leq T_1^+(t)$ and $T_R(t) = T_R^+(t)$.

In event E1, we have, since $N_r(S(t)) \leq N_r(S^+(t))$ and $T_A(t) = T_1^+(t)$, that $\xi_A(t) \leq \xi_1^+(t)$ (Relation (a)). When $\xi_1^+(t) = 0$, there are no tokens in both systems, so no job will be accepted in either system. When $\xi_1^+(t) > 0$, the job in system S^+ will definitely be accepted (but not necessarily in system S), so that we still have $T_A(t) \leq T_1^+(t)$ after the arrival. Similarly, in event E2, $\xi_R(t) \leq \xi_R^+(t)$ (Relation (b)). After the arrival, we still have $T_R(t) \leq T_R^+(t)$. With this reasoning, we have proven that (2.9) is true for all t. Since the number of accepted jobs at time t is $T_A(t) + T_R(t)$, more jobs have been accepted in system S^+, which proves that the blocking probability of system S^+ is smaller.

Worse system. We can show using coupling that the blocking probability of S^- is greater than the blocking probability of S. We will use the superscript $-$ to denote a variable of the worse system. We will only provide a sketch of the proof as it is similar to the previous coupling. The counterpart of (2.9) will now be

$$T_r^-(t) \leq T_r(t) \text{ for all } 1 \leq r \leq R.$$

This yields $\xi_r^-(t) \leq \xi_r(t)$ if $T_r^-(t) = T_r(t)$. All jobs accepted in system S^- will also be accepted in S, because there are at least as many tokens at the dispatcher in system S. So, after any event, $T_r^-(t) \leq T_r(t)$ will still be true. Finally, that means that the number of accepted jobs is less in system S^-, so that the blocking probability is larger.

2.4 Fluid limit in the blocking scenario

We now turn to the fluid-limit analysis and start with the blocking scenario. The concept of fluid limits was introduced in Section 1.3.2. We consider a sequence of systems indexed by the total number of servers N. Denote by $x_0^N(t) = \frac{1}{N} X_0^N(t)$ the fraction of busy servers and by $x_r^N(t) = \frac{1}{N} X_r^N(t)$ the normalized number of tokens held by dispatcher r in the N-th system at time t. Further define $x^N(t) = (x_0^N(t), x_1^N(t) \ldots, x_R^N(t))$ and assume that $x^N(0) \to x^\infty$ as $N \to \infty$, with $\sum_{i=0}^R x_i^\infty = 1$. Then any weak limit $x(t)$ of the sequence $\{x^N(t)\}_{t \geq 0}$ as $N \to \infty$ is called a fluid limit.

The fluid limit $x(t)$ in the blocking scenario with Enhancements 2.1 and 2.2 in place satisfies the set of differential equations

$$\frac{\mathrm{d}x_0(t)}{\mathrm{d}t} = \sum_{r=1}^R z_r(t) - x_0(t), \tag{2.10}$$

$$\frac{\mathrm{d}x_r(t)}{\mathrm{d}t} = \beta_r x_0(t) + \nu\left(\frac{1-x_0(t)}{R} - x_r(t)\right) - z_r(t), \tag{2.11}$$

with

$$z_r(t) = \alpha_r\lambda - \left[\alpha_r\lambda - \beta_r x_0(t) - \nu\frac{1-x_0(t)}{R}\right]^+ \mathbb{1}\{x_r(t)=0\}, \tag{2.12}$$

where $[\cdot]^+ = \max\{\cdot, 0\}$ and initial condition $x(0) = x^\infty$.

The above set of fluid-limit equations may be interpreted as follows. The term $z_r(t)$ represents the (scaled) rate at which dispatcher r uses tokens and forwards incoming jobs to idle servers at time t. (2.12) reflects that the latter rate equals the job arrival rate $\alpha_r\lambda$, unless the fraction of tokens held by dispatcher r is zero ($x_r(t) = 0$), and the rate $\beta_r x_0(t) + \nu(1 - x_0(t))/R$ at which it receives tokens from idle servers or through the exchange mechanism is less than the job arrival rate. (2.10) states that the rate of change in the fraction of busy servers is the difference between the aggregate rate $\sum_{r=1}^{R} z_r(t)$ at which the various dispatchers use tokens and forward jobs to idle servers, and the rate $x_0(t)$ at which jobs are completed and busy servers become idle. (2.11) captures that the rate of change of the fraction of tokens held by dispatcher r is the balance of the rate $\beta_r x_0(t) + \nu(1 - x_0(t))/R$ at which it receives tokens from idle servers or through the exchange mechanism, and the rate $z_r(t) + \nu x_r(t)$ at which it uses tokens and forwards jobs to idle servers or releases tokens through the exchange mechanism.

Figure 2.4 shows the exact and simulated fluid-limit trajectories. We observe that the simulation results closely match the fluid-limit dynamics. We further note that in the long run only dispatcher 2 with the lower arrival rate holds a strictly positive fraction of the tokens, corroborating Theorem 1.

2.4.1 Derivation of fluid-limit equations

Similarly to [HK94; PTW07], (2.10)–(2.12) can be derived by describing the system with N servers in terms of elementary stochastic processes and random variables. Let $A_y^1, \ldots, A_y^R$ and S_y be independent Poisson processes with rate y and define random variables Φ_n, which are $1, \ldots, R$ with probability $1/R$. We obtain the following system description.

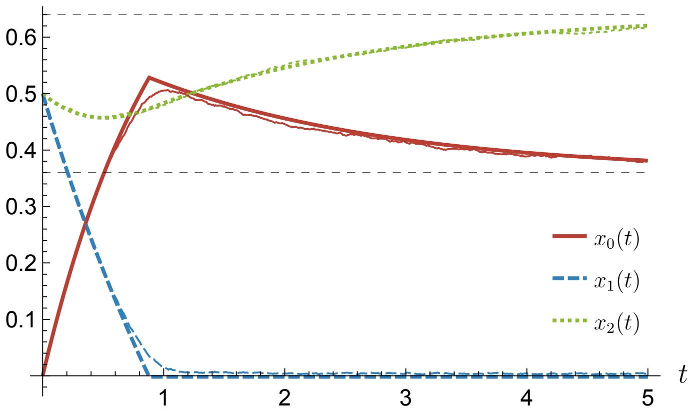


Figure 2.4: Exact and simulated fluid-limit trajectories $x_i(t)$ for $R = 2$ dispatchers, $\lambda = 0.9$ and $\alpha_1 = 0.8$. Averaged sample paths for $N = 100$ servers are represented by dashed lines, and closely match the fluid-limit dynamics.

$$X_0^N(t) = X_0^N(0) + \sum_{r=1}^{R} A^r_{\alpha_r \lambda N}\left(\int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s\right) - S_1\left(\int_0^t X_0^N(s)\mathrm{d}s\right), \tag{2.13}$$

$$X_r^N(t) = X_r^N(0) + \sum_{n=1}^{S_1\left(\int_0^t X_0^N(s)\mathrm{d}s\right)} \mathbb{1}\{\Phi_n = r\} - A^r_{\alpha_r \lambda N}\left(\int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s\right) \tag{2.14}$$

for $1 \le r \le R$. Dividing (2.13) and (2.14) by N gives:

$$\bar{M}_0^N(t) = \bar{M}_0^N(0) + \frac{1}{N}\sum_{r=1}^{R} A^r_{\alpha_r \lambda N}\left(\int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s\right) - \frac{1}{N}S_1\left(N\int_0^t \bar{M}_0^N(s)\mathrm{d}s\right), \tag{2.15}$$

$$\bar{M}_r^N(t) = \bar{M}_r^N(0) + \frac{1}{N}\sum_{n=1}^{S_1\left(\int_0^t X_0^N(s)\mathrm{d}s\right)} \mathbb{1}\{\Phi_n = r\} - \frac{1}{N}A^r_{\alpha_r \lambda N}\left(\int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s\right) \tag{2.16}$$

for $1 \le r \le R$.

We will rewrite these expressions, by observing that some terms converge to simpler functions as $N \to \infty$. We state Lemmas 2.1 and 2.2, in which the convergence is almost sure.

Lemma 2.1.

$$\frac{1}{N}\left[\sum_{n=1}^{S_1\left(\int_0^t X_0^N(s)\mathrm{d}s\right)} \mathbb{1}\{\Phi_n = r\} - \frac{1}{R}S_1\left(\int_0^t X_0^N(s)\mathrm{d}s\right)\right] \to 0 \quad \text{as} \quad N \to \infty. \tag{2.17}$$

Lemma 2.2.

$$\frac{1}{n}\left[A_{xn}^r(t) - xnt\right] \to 0 \quad \text{and} \quad \frac{1}{n}\left[A_x^r(nt) - xnt\right] \to 0 \quad \text{as} \quad n \to \infty.$$

We omit formal proofs but provide intuition. We recognize a marked point process in Lemma 2.1, in which $\mathbb{P}\{\Phi_n = r\} = 1/R$. We view the first term as a binomially distributed random variable, so that the second term is its mean. Since $S_1(\int_0^t X_0^N(s)\mathrm{d}s) \to \infty$, we apply the central limit theorem. We have that the left-hand side of (2.17) multiplied by $\sqrt{N}$ converges to a normally distributed random variable. This shows that the left-hand side as in (2.17) converges to 0.

Similarly in Lemma 2.2, we view $A_{xn}^r(t)$ and $A_x^r(nt)$ as a Poisson random variable with mean xnt, from which we subtract its mean. The central limit theorem gives that both expressions multiplied by $\sqrt{n}$ converge to a normally distributed random variable, so that the expressions themselves converge to 0.

Applying Lemmas 2.1 and 2.2 to (2.15) and (2.16) gives (for large N)

$$\begin{aligned} \bar{M}_0^N(t) &= \bar{M}_0^N(0) + \sum_{r=1}^R \alpha_r\lambda \int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s - \int_0^t \bar{M}_0^N(s)\mathrm{d}s, \\ \bar{M}_r^N(t) &= \bar{M}_r^N(0) + \frac{1}{R}\int_0^t \bar{M}_0^N(s)\mathrm{d}s - \alpha_r\lambda\int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s \text{ for } 1 \le r \le R. \end{aligned} \tag{2.18}$$

Finally, we find an alternate expression for $F(t) := \int_0^t \mathbb{1}\{X_r^N(s) > 0\}\mathrm{d}s$, which stands for the amount of time that there was a positive number of tokens at dispatcher r between times 0 and t. We analyze the derivative of $F(t)$ denoted by $f(t) = \mathbb{1}\{X_r^N(t) > 0\}$ and express it in terms of $\bar{M}_i(t)$. We first note that if $\bar{M}_r(t) > 0$, then surely $X_r(t) > 0$ so that $f(t) = 1$. Vice versa however, $X_r^N(t) = a$ (constant) means that as $N \to \infty$, $\bar{M}_r(t) = 0$. We therefore carefully analyze $f(t)$ in case $\bar{M}_r^N(t) = 0$.

We introduce the vector $L^N(t)$, for which we have $L_i^N(t) = \frac{1}{t}\int_0^t \mathbb{1}\{X_r^N(s) = i\}\mathrm{d}s$ for $i = 0, 1, \ldots N$. Intuitively, $L_i^N(t)$ is the fraction of time that dispatcher r has

i tokens in a system with N servers. To simplify notation, we fix time t. We will consider $L^N(t)$ where $N \to \infty$; we define the limiting probability $\pi(i) := \lim_{N\to\infty} L_i^N(t)$. Since $L^N(t)$ has length $N+1$, $\pi(i)$ exists for all natural numbers i, as well as for 0 and for ∞. We still have $\sum_i \pi_i(t) = 1$. In some cases we find that there exist values of i such that $\pi(i) > 0$, in other cases $\pi(i) = 0$ for every single value of i. Note that the latter property is a well-known property of many measures, for example the Lebesgue measure.

We will now establish an expression for $\pi(\cdot)$ in a heuristic manner. We can compute $\pi(\cdot)$ by defining an M/M/1 queue with arrival rate $X_0^N(t)/R$ and departure rate $\alpha_r \lambda N$. Since the rates are scaled by N, for large N the system will be in equilibrium at any time $t > 0$. For equilibrium probabilities, the system is equivalent to a system with rates $\bar{M}_0^N(t)/R$ and $\alpha_r\lambda$. When $\bar{M}_0^N(t)/R \geq \alpha_r\lambda$, we find that the queue length is a random walk with transition probabilities $p_{i,i+1} > p_{i,i-1}$, so that state 0 is a transient state (or null-recurrent for equality). In this case we will find that $\pi(0) = 0$. When $\bar{M}_0^N(t)/R \leq \alpha_r\lambda$, state 0 is a recurrent state, with equilibrium probability

$$1 - \frac{\bar{M}_0^N(t)/R}{\alpha_r\lambda}.$$

Since we are interested in the fraction of time that the number of tokens is strictly greater than 0, we consider

$$1 - \pi(0) = \min\left\{1, \frac{\bar{M}_0^N(t)/R}{\alpha_r\lambda}\right\}.$$

These observations lead to the assertion that for large N

$$f(t) = \mathbb{1}\{X_r^N(t) > 0\} = \begin{cases} \min\left\{1, \frac{\bar{M}_0^N(t)/R}{\alpha_r\lambda}\right\} & \bar{M}_r(t) = 0, \\ 1 & \bar{M}_r(t) > 0. \end{cases} \tag{2.19}$$

We substitute (2.19) in (2.18) and omit the parameter N, since all substitutions rely on large N. We then arrive at

$$\bar{M}_0(t) = \bar{M}_0(0) + \sum_{r=1}^{R} t\alpha_r\lambda f(t) - \int_0^t \bar{M}_0(s)\mathrm{d}s,$$

$$\bar{M}_r(t) = \bar{M}_r(0) + \frac{1}{R}\int_0^t \bar{M}_0(s)\mathrm{d}s - t\alpha_r\lambda f(t) \text{ for } 1 \leq r \leq R.$$

Taking derivatives with respect to t gives the set of ODEs as described in (2.10)–(2.12).

2.4.2 Fixed-point analysis

In order to determine the fixed point(s) x^*, we set $\mathrm{d}x_i(t)/\mathrm{d}t = 0$ for all $i = 0, 1, \ldots, R$, and obtain

$$x_0^* = \sum_{r=1}^{R} z_r^*, \tag{2.20}$$

$$z_r^* = \beta_r x_0^* + \nu \left(\frac{1 - x_0^*}{R} - x_r^* \right), \tag{2.21}$$

and

$$z_r^* = \alpha_r \lambda - \left[\alpha_r \lambda - \beta_r x_0^* - \nu \frac{1 - x_0^*}{R} \right]^+ \mathbb{1}\{x_r^* = 0\}. \tag{2.22}$$

Without proof, we assume that the many-server ($N \to \infty$) and stationary ($t \to \infty$) limits commute, so that x_0^* is also the limit of the mean fraction of busy servers in stationarity (see Section 1.3.2). Without proof we couple the fixed point of the fluid limit with the stationary blocking probability. Because of Little's law, the limit B of the blocking probability satisfies

$$x_0^* = \lambda(1 - B). \tag{2.23}$$

This in particular implies that $x_0^* = \lambda$ leads to $B = 0$: vanishing blocking.

Basic JIQ scheme. We first consider the basic JIQ scheme, i.e., $\beta_r = 1/R$ for all $r = 1, \ldots, R$ and $\nu = 0$. (2.21) and (2.22) yield

$$\frac{x_0^*}{R} = z_r^* = \alpha_r \lambda - \left[\alpha_r \lambda - \frac{x_0^*}{R} \right]^+ \mathbb{1}\{x_r^* = 0\},$$

or equivalently,

$$\alpha_r \lambda - \frac{x_0^*}{R} = \left[\alpha_r \lambda - \frac{x_0^*}{R} \right]^+ \mathbb{1}\{x_r^* = 0\}. \tag{2.24}$$

Now let $\mathscr{I} = \{r : \alpha_r = \alpha_R\}$ be the index set of the least-loaded dispatchers. (2.24) forces $x_r^* = 0$ for all $r \notin \mathscr{I}$.

We now distinguish two cases, depending on whether or not $x_r^* = 0$ for all $r \in \mathscr{I}$ as well. If that is the case, then we must have $x_0^* = 1$, and $\alpha_R \lambda \geq x_0^*/R$, i.e.,

$\lambda \geq 1/(R\alpha_R)$. Otherwise, we must have $\alpha_R\lambda = x_0^*/R$, i.e., $x_0^* = R\alpha_R\lambda$, so $x_0^* \leq 1$ forces $\lambda \leq 1/(R\alpha_R)$.

In conclusion, we have $x_0^* = \min\{R\alpha_R\lambda, 1\}$. When $\lambda \geq 1/(R\alpha_R)$ so that $x_0^* = 1$, it must be the case that $x_r^* = 0$ for all $r = 1,\ldots,R$. When $\lambda < 1/(R\alpha_R)$ so that $x_0^* < 1$, any vector x^* with $x_r^* = 0$ for all $r \notin \mathscr{I}$ and $\sum_{r\in\mathscr{I}} x_r^* = 1 - x_0^*$ is a fixed point. In particular, for equal dispatcher loads, i.e., $\alpha_r = 1/R$ for all $r = 1,\ldots,R$, so that $\mathscr{I} = \{1,\ldots,R\}$, we have $x_r^* = 0$ for all $r = 1,\ldots,R$ when $\lambda \geq 1$, while any vector x^* with $\sum_{r=1}^{R} x_r^* = 1 - \lambda$ is a fixed point when $\lambda < 1$.

We use (2.23) to find $B = 1 - \frac{1}{\lambda}\min\{R\alpha_R\lambda, 1\} = \max\{1 - R\alpha_R, 1 - 1/\lambda\}$, which agrees with Theorem 2.1.

In Table 2.1 we compare the fluid-limit approximations for the blocking probability with the exact formula from the Jackson network representation and simulation results for various numbers of servers. Table 2.1 shows that the Jackson network analysis agrees with the simulation results. Furthermore, the more symmetric the loads, the lower the blocking probability, which is consistent with Theorem 2.1. Also, the fluid-limit approximation is highly accurate, even for a fairly small number of servers.

Table 2.1: Blocking probabilities for $\lambda = 0.9$ and $\alpha_1 = 0.8$ or $\alpha_1 = 0.6$.

	$\lambda = 0.9, \alpha_1 = 0.8$		$\lambda = 0.9, \alpha_1 = 0.6$	
N	Jackson	simulation	Jackson	simulation
10	0.6021	0.6032	0.3201	0.3205
20	0.6000	0.6006	0.2545	0.2552
50	0.6000	0.6004	0.2092	0.2095
100	0.6000	0.6007	0.2006	0.2010
fluid ($N = \infty$)	0.6000	-	0.2000	-

2.4.3 Enhancements

We now examine the behavior of the system for Enhancements 2.1 and 2.2, and show that they can achieve asymptotically zero blocking for any $\lambda \leq 1$ and suitable parameter values as identified in Theorem 2.3. In light of (2.23) it suffices to show that $x_0^* = \lambda$ for both enhancements.

Consider Enhancement 2.1; $\beta_r = \alpha_r$ and $\nu = 0$. (2.21) and (2.22) give $\lambda - x_0^* = [\lambda - x_0^*]^+ \mathbb{1}\{x_r^* = 0\}$ for all r (this shows $x_0^* \leq \lambda$). Assume that $x_0^* < \lambda$. Then, $x_r^* = 0$

for all r, which results in $x_0^* = 1$, since $\sum_{i=0}^R x_i^* = 1$. This contradicts $x_0^* \le \lambda < 1$, so that $x_0^* = \lambda$.

Notice that any point for which $x_0^* = \lambda$ and for which $\sum_{i=0}^R x_i^* = 1$, results in $z_r^* = \alpha_r\lambda$ which is in accordance with (2.20)–(2.22), and therefore is a fixed point.

We next consider Enhancement 2.2 with $\beta_r = 1/R$ and $\nu \ge \frac{\lambda}{1-\lambda}(R\alpha_1 - 1)$. We use (2.20) and (2.22) twice, first they give $x_0^* \le \lambda$, so that $x_0^* = \lambda - \varepsilon$ for some $\varepsilon \ge 0$. Second, $z_r^* = \alpha_r\lambda$ if the term in brackets in (2.22) is non-positive. Otherwise,

$$\begin{aligned}
z_r^* &\ge \alpha_r\lambda - \left[\alpha_r\lambda - \frac{x_0^*}{R} - \nu\frac{1-x_0^*}{R}\right] \\
&\ge \alpha_r\lambda - \alpha_r\lambda + \frac{\lambda-\varepsilon}{R} + \frac{\lambda}{1-\lambda}(R\alpha_1 - 1)\frac{1-\lambda}{R} + \nu\frac{\varepsilon}{R} \\
&\ge \alpha_r\lambda - \frac{\varepsilon}{R}(1-\nu),
\end{aligned}$$

which by (2.20) gives $\lambda - \varepsilon = x_0^* = \sum_{r=1}^R z_r^* \ge \lambda - \varepsilon(1-\nu)$, so that $x_0^* = \lambda$.

Figure 2.5 displays the blocking probability as $N \to \infty$ for the system with both enhancements. Since $\alpha_1 = 0.7$, we have that $\beta_1 = 0.7$ is optimal. The blocking probability decreases as β_1 approaches α_1 and as ν increases. For $\nu > 0$, it suffices to choose β_1 close to α_1, which implies that it is not necessary to know the exact loads, for the enhancements to be effective.

2.5 Fluid limit in the queueing scenario

We now proceed to the queueing scenario (with $\lambda < 1$ for stability). As before, we consider a sequence of systems indexed by the total number of servers N. Denote by $y_i^N(t) = \frac{1}{N}Y_i^N(t)$ the fraction of servers with i jobs and by $x_r^N(t) = \frac{1}{N}X_r^N(t)$ the normalized number of tokens held by dispatcher r in the N-th system at time t, $r = 1,\dots,R$. Further define $u^N(t) = (y^N(t), x_1^N(t)\dots, x_R^N(t))$, with $y^N(t) = (y_i^N(t))_{i\ge0}$, and assume that $u^N(0) \to u^\infty$ as $N \to \infty$, with $\sum_{i=1}^\infty y_i^\infty + \sum_{r=1}^R x_r^\infty = 1$. Then any weak limit $u(t)$ of the sequence $\{u^N(t)\}_{t\ge0}$ as $N \to \infty$ is called a fluid limit.

The fluid limit $u(t)$ in the queueing scenario with Enhancements 2.1 and 2.2 in place obeys the set of differential equations

$$\frac{\mathrm{d}y_0(t)}{\mathrm{d}t} = y_1(t) - \lambda_1(t) - \lambda_2(t)y_0(t), \tag{2.25}$$

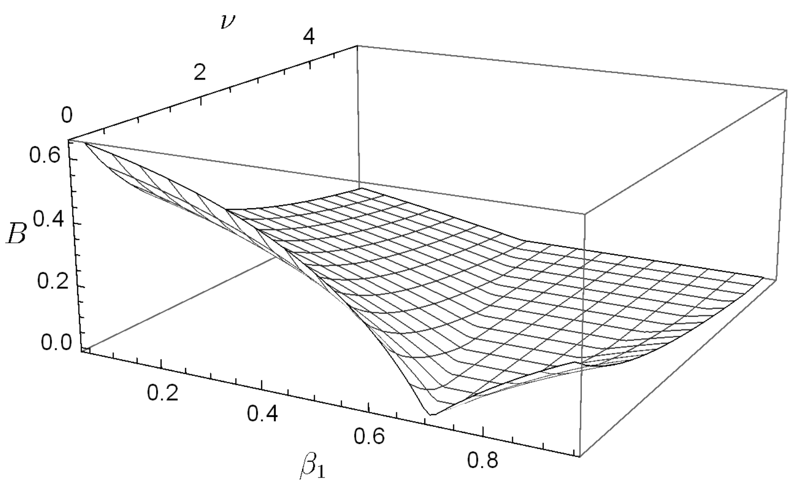


Figure 2.5: Blocking probability B in the limit for $R = 2$, $\lambda = 0.9$ and $\alpha_1 = 0.7$, for different values of β_1 and ν.

$$\begin{aligned}\frac{\mathrm{d}y_i(t)}{\mathrm{d}t} &= \lambda_1(t)\mathbb{1}\{i=1\} + \lambda_2(t)y_{i-1}(t) \\ &\quad + y_{i+1}(t) - y_i(t) - \lambda_2(t)y_i(t) \text{ for all } i \geq 1,\end{aligned} \tag{2.26}$$

$$\frac{\mathrm{d}x_r(t)}{\mathrm{d}t} = \beta_r y_1(t) + \nu\left(\frac{y_0(t)}{R} - x_r(t)\right) - z_r(t) - \lambda_2(t)x_r(t), \tag{2.27}$$

with

$$z_r(t) = \alpha_r\lambda - \left[\alpha_r\lambda - \beta_r y_1(t) - \nu\frac{y_0(t)}{R}\right]^+ \mathbb{1}\{x_r(t) = 0\},$$

$$\lambda_1(t) = \sum_{r=1}^{R} z_r(t), \quad \lambda_2(t) = \lambda - \lambda_1(t), \tag{2.28}$$

and initial condition $u(0) = u^\infty$.

The above set of fluid-limit equations may be interpreted as follows. Similarly as in the blocking scenario, the term $z_r(t)$ represents the (scaled) rate at which dispatcher r uses tokens and forwards incoming jobs to idle servers at time t. Accordingly, $\lambda_1(t)$ is the aggregate rate at which dispatchers use tokens to forward jobs to (guaranteed) idle servers at time t, while $\lambda_2(t)$ is the aggregate rate at which jobs are forwarded to randomly selected servers (which may or may not be idle). (2.25) reflects that the rate of change in the fraction of idle servers is the difference between the aggregate rate $y_0(t)$ at which jobs are completed by servers with one job, and the rate $\lambda_1(t)$ at which dispatchers use tokens to forward jobs to idle servers plus the rate $\lambda_2(t)y_0(t)$ at which jobs are forward to randomly selected servers that happen to be idle. (2.26) states that the rate of change in the fraction of servers with i jobs is the balance of the rate $\lambda_2(t)y_{i-1}(t)$ at which jobs are forwarded to randomly selected servers with $i-1$ jobs plus the aggregate rate $y_{i+1}(t)$ at which jobs are completed by servers with $i+1$ jobs, and the rate $\lambda_2(t)y_i(t)$ at which jobs are forwarded to randomly selected servers with i jobs plus the aggregate rate $y_i(t)$ at which jobs are completed by servers with i jobs. In case $i=1$, the rate at which dispatchers use tokens to forward jobs to idle servers should be included as additional positive term.

(2.27) is similar to (2.11), where the additional fourth term captures the rate at which tokens are revoked when jobs are forwarded to randomly selected servers that happen to be idle.

2.5.1 Fixed-point analysis

In order to determine the fixed point(s) u^*, we set $\mathrm{d}y_i(t)/\mathrm{d}t = 0$ for all $i \geq 0$, and $\mathrm{d}x_r(t)/\mathrm{d}t = 0$ for all $r = 1, \ldots, R$. We obtain

$$y_1^* = \lambda_1^* + \lambda_2^* y_0^*, \quad (1+\lambda_2^*)y_i^* = \lambda_2^* y_{i-1}^* + y_{i+1}^* \text{ for all } i \geq 2. \tag{2.29}$$

Solving (2.29) gives

$$y_0^* = 1-\lambda, \quad y_k^* = \lambda(1-\lambda_2^*)\left(\lambda_2^*\right)^{k-1} \text{ for all } k \geq 1.$$

Thus the mean number of jobs at a server is

$$\sum_{k=1}^{\infty} k y_k^* = \sum_{k=1}^{\infty} k\lambda(1-\lambda_2^*)\left(\lambda_2^*\right)^{k-1} = \frac{\lambda}{1-\lambda_2^*}.$$

As for the blocking scenario, we use the fixed point of the fluid limit without proof as indication or the stationary waiting time in the queueing scenario.

Little's law then gives

$$\sum_{k=1}^{\infty} k y_k^* = \lambda(\mathbb{E}[W]+1), \tag{2.30}$$

where the left-hand side represents the mean number of jobs at a server in the many-server limit. We use (2.30) to obtain

$$\mathbb{E}[W] = \frac{\sum_{k=1}^{\infty} k y_k^*}{\lambda} - 1 = \frac{\lambda_2^*}{1-\lambda_2^*}, \tag{2.31}$$

which shows vanishing wait in case $\lambda_2^* = 0$.

We also obtain the following equations for the fixed point:

$$z_r^* = \beta_r \lambda(1-\lambda_2^*) - \lambda_2^* x_r^* + \nu\left(\frac{1-\lambda}{R} - x_r^*\right), \tag{2.32}$$

$$z_r^* = \alpha_r\lambda - \left[\alpha_r\lambda - \beta_r\lambda(1-\lambda_2^*) - \nu\frac{1-\lambda}{R}\right]^+ \mathbb{1}\{x_r^* = 0\}, \tag{2.33}$$

and

$$\lambda_1^* = \sum_{r=1}^{R} z_r^*. \tag{2.34}$$

We define

$$q_r^* = z_r^*\Big|_{x_r^*=0} = \beta_r\lambda(1-\lambda_2^*) + \nu\frac{1-\lambda}{R}. \tag{2.35}$$

(2.32) implies $z_r^* \le q_r^*$ and (2.33) leads to $z_r^* \le \alpha_r\lambda$, $x_r^* = 0 \Rightarrow z_r^* = q_r^*$ and $x_r^* > 0 \Rightarrow z_r^* = \alpha_r\lambda$, yielding $z_r^* = \min\{q_r^*, \alpha_r\lambda\}$ and thus

$$\lambda_1^* = \sum_{r=1}^{R} \min\{q_r^*, \alpha_r\lambda\}. \tag{2.36}$$

Basic JIQ scheme. In case $\beta_r = 1/R$ and $\nu = 0$, (2.36) can be rewritten to

$$\lambda_2^* = \sum_{r=1}^{R} \left[\alpha_r\lambda - \frac{\lambda(1-\lambda_2^*)}{R}\right]^+, \tag{2.37}$$

and by further calculations, since α_r is decreasing in r, to the expression for λ_2^* in Theorem 2.2.

In Table 2.2 we compare the fluid-limit approximations for the mean-waiting time with simulation figures for various numbers of servers. Table 2.2 shows that the fluid-limit analysis agrees with the simulation results, although the number of servers needs to be larger and simulations need to be longer than in the blocking scenario for extremely high accuracy to be observed. Similarly to Table 2.1, the more symmetric the loads, the better the performance and the lower the mean waiting time, which is in line with Theorem 2.2.

Table 2.2: Mean waiting time for $\lambda = 0.9$ and $\alpha_1 = 0.8$ or $\alpha_1 = 0.6$.

N	$\lambda = 0.9, \alpha_1 = 0.8$ simulation	$\lambda = 0.9, \alpha_1 = 0.6$ simulation
10	2.5824	2.1234
20	1.7349	1.1386
50	1.1704	0.5001
100	1.0173	0.2981
200	0.9764	0.2207
500	0.9631	0.1983
1000	0.9599	0.1962
fluid ($N = \infty$)	0.9643	0.1957

2.5.2 Enhancements

We examine the behavior of the system for Enhancements 2.1 and 2.2 and show that they can achieve asymptotically zero waiting for any $\lambda < 1$ and suitable parameter values as identified in Theorem 2.3. In view of (2.31) it suffices to show that $\lambda_2^* = 0$ for both enhancements. We first consider Enhancement 2.1 in which $\alpha_r = \beta_r$ for all r and $\nu = 0$. (2.35) gives $q_r^* - \alpha_r\lambda = -\beta_r\lambda\lambda_2^* \leq 0$ for all r and $q_r^* = \beta_r\lambda(1 - \lambda + \lambda_1^*)$. We obtain

$$\lambda_1^* = \sum_{r=1}^{R} q_r^* = \lambda(1 - \lambda + \lambda_1^*),$$

which has a unique solution $\lambda_1^* = \lambda$, so that $\lambda_2^* = 0$.

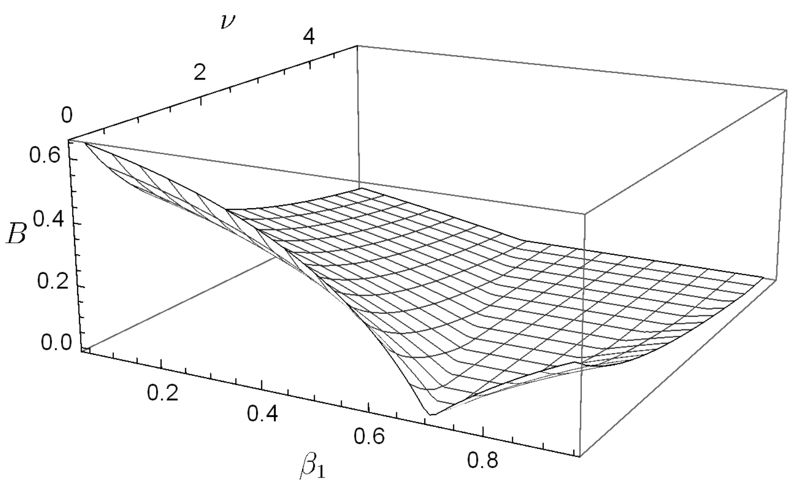


Figure 2.6: Mean waiting time $\mathbb{E}[W]$ in the limit for $R = 2$, $\lambda = 0.9$ and $\alpha_1 = 0.7$, for different values of β_1 and ν.

Next, we turn to Enhancement 2.2 where $\nu \geq \frac{\lambda}{1-\lambda}(R\alpha_1 - 1)$ and $\beta_r = 1/R$. If the term in brackets in (2.33) is non-positive, $z_r^* = \alpha_r\lambda$. Otherwise,

$$\begin{aligned} z_r^* &\geq \alpha_r\lambda - \left[\alpha_r\lambda - \frac{\lambda(1-\lambda_2^*)}{R} - \nu\frac{1-\lambda}{R}\right] \\ &\geq \alpha_r\lambda - \alpha_r\lambda + \frac{\lambda - \lambda\lambda_2^*}{R} + \frac{\lambda}{R}(R\alpha_1 - 1) \geq \alpha_r\lambda - \frac{\lambda\lambda_2^*}{R}, \end{aligned}$$

which by (2.34) gives $\lambda - \lambda_2^* = \lambda_1^* \geq \lambda - \lambda\lambda_2^*$, and since $\lambda < 1$, we obtain $\lambda_2^* = 0$.

Figure 2.6 displays the mean waiting time as $N \to \infty$ for the system with both enhancements. Similarly to Figure 2.5, we can greatly improve the performance by tuning β and ν. Again $\alpha_1 = 0.7$, so that $\beta_1 = 0.7$ is the best choice. The mean waiting time decreases as β_1 approaches α_1, or as the rate ν increases. Exact knowledge of the arrival rates is not required, and a rough approximation of β_1 and a small value of ν are sufficient for the mean waiting time to vanish.

2.6 Conclusion

We examined the performance of the JIQ scheme in large-scale systems with several possibly heterogeneous dispatchers. We used product-form representations and fluid limits to show that the basic JIQ scheme fails to deliver zero blocking and wait for any asymmetric dispatcher loads, even for arbitrarily low overall load. Remarkably, it is the least-loaded dispatcher that throttles tokens and leaves idle servers stranded, thus acting as bottleneck.

In order to counter the performance degradation for asymmetric dispatcher loads, we introduced two extensions of the basic JIQ scheme where tokens are either distributed non-uniformly or occasionally exchanged among the various dispatchers. We proved that these extensions can achieve zero blocking and wait in the many-server limit, for any subcritical overall load and arbitrarily skewed load profiles. Extensive simulation experiments corroborated these results, indicating that they apply even in moderately sized systems.

It is worth emphasizing that the proposed enhancements involve no or constant additional communication overhead per job, and hence retain the scalability of the basic JIQ scheme. The algorithms do rely on suitable parameter settings, and it would be of interest to develop learning techniques for that.

While we allowed the dispatchers to be heterogeneous, we assumed all the servers to be statistically identical, and the service requirements to be exponentially distributed. As noted earlier, Theorem 2.1 in fact holds for non-exponential service requirement distributions as well. It could be interesting to extend Theorems 2.2 and 2.3 to possibly non-exponential service requirement distributions.

2.A Proof of Proposition 2.1

We write (2.7) as

$$B(2,N,\lambda,\alpha_1,\alpha_2,\beta_1,\beta_2) = (\gamma_1-\gamma_2)\frac{1+\frac{\alpha_2}{\alpha_1}\frac{\gamma_1}{\gamma_2}Z(N,\lambda,\gamma_1,\gamma_2)}{\frac{\gamma_1}{\alpha_1}-\frac{\gamma_1}{\alpha_1}Z(N,\lambda,\gamma_1,\gamma_2)} \tag{2.38}$$

with

$$Z(N,\lambda,\gamma_1,\gamma_2) = \frac{\gamma_2}{\gamma_1}\frac{\sum_{n=0}^{N}\left(\frac{\gamma_2}{\gamma_1}\right)^{N-n}\frac{(2\gamma_2\lambda N)^n}{n!}}{\sum_{n=0}^{N}\frac{(2\gamma_2\lambda N)^n}{n!}}. \tag{2.39}$$

Lemma 2.3. *Assume* $\gamma_1 > \gamma_2$.

(a) For $2\gamma_2\lambda \le 1$, $\lim_{N\to\infty} Z(N,\lambda,\gamma_1,\gamma_2) = 0$.

(b) For $2\gamma_2\lambda > 1$,

$$\lim_{N\to\infty} Z(N,\lambda,\gamma_1,\gamma_2) = \frac{\gamma_2}{\gamma_1}\frac{2\gamma_2\lambda - 1}{2\gamma_2\lambda - \frac{\gamma_2}{\gamma_1}}.$$

To prove Lemma 2.3 we first state several auxiliary results.

Lemma 2.4. *For* $k \ge 0$ *and* $\eta > 0$,

$$T(N) := \frac{(N-k)!}{N^\eta (N-\eta-k)!} \to 1 \quad \text{as} \quad N \to \infty.$$

Proof. Note that $T(N) = \frac{N-k}{N}\cdot\frac{N-k-1}{N}\cdot\ldots\cdot\frac{N-\eta-k+1}{N}$ and hence

$$\left(\frac{N-\eta-k}{N}\right)^\eta \le T(N) \le 1.$$

Since $\frac{N-\eta-k}{N} \to 1$, we have $\left(\frac{N-\eta-k}{N}\right)^\eta \to 1$, which shows that $T(N) \to 1$. $\square$

Lemma 2.5. *For* $x \le 1$ *and* $k \ge 0$,

$$\frac{\frac{(xN)^{N-k}}{(N-k)!}}{\sum_{n=0}^N \frac{(xN)^n}{n!}} \to 0 \quad \text{as} \quad N \to \infty.$$

Proof. First assume $x < 1$. Let $\varepsilon > 0$ and define $\bar\varepsilon := \lfloor 1/\varepsilon \rfloor$. Fix $\bar N$ such that $\frac{\bar N-k-\bar\varepsilon}{\bar N} > x$. Note that such an $\bar N$ exists because $x < 1$ and $\frac{\bar N-k-\bar\varepsilon}{\bar N}$ tends to 1 for large $\bar N$. For $N > \bar N$, we find $\frac{xN}{N-k-\bar\varepsilon} < 1$. For all $i \le \bar\varepsilon$,

$$\frac{(xN)^{N-k-i}}{(N-k-i)!} < \frac{\frac{(xN)^{N-k-i}}{(N-k-i)!}}{\frac{xN}{N-k-\bar\varepsilon}} \le \frac{\frac{(xN)^{N-k-i}}{(N-k-i)!}}{\frac{xN}{N-k-i}} = \frac{(xN)^{N-k-i-1}}{(N-k-i-1)!},$$

which leads to

$$\frac{(xN)^{N-k}}{(N-k)!} < \frac{(xN)^{N-k-1}}{(N-k-1)!} < \frac{(xN)^{N-k-2}}{(N-k-2)!} < \ldots < \frac{(xN)^{N-k-\bar\varepsilon-1}}{(N-k-\bar\varepsilon-1)!}.$$

We find

$$\sum_{n=0}^{N} \frac{(xN)^n}{n!} > \sum_{n=N-k-\bar{\varepsilon}-1}^{N-k} \frac{(xN)^n}{n!} > (\bar{\varepsilon}+1)\frac{(xN)^{N-k}}{(N-k)!}$$

so that for all $N > \bar{N}$,

$$\frac{\frac{(xN)^{N-k}}{(N-k)!}}{\sum_{n=0}^{N} \frac{(xN)^n}{n!}} < \frac{\frac{(xN)^{N-k}}{(N-k)!}}{(\bar{\varepsilon}+1)\frac{(xN)^{N-k}}{(N-k)!}} = \frac{1}{\bar{\varepsilon}+1} < \varepsilon.$$

Next consider $x = 1$. Let $\varepsilon > 0$ be given. Fix $\hat{\varepsilon} = \lfloor 1/\varepsilon \rfloor + 1$. Fix $\bar{N}$ so that for all $N > \bar{N}$,

$$\frac{(N-k)!}{N^n(N-n-k)!} > 1 - \frac{1}{\hat{\varepsilon}+1},$$

for $n = 0, \dots, \hat{\varepsilon}$ using Lemma 2.4. Then we find

$$\frac{\frac{N^{N-k}}{(N-k)!}}{\sum_{n=0}^{N} \frac{N^n}{n!}} \leq \frac{\frac{N^{N-k}}{(N-k)!}}{\sum_{n=N-k-\hat{\varepsilon}}^{N-k} \frac{N^n}{n!}} = \frac{\frac{N^{N-k}}{(N-k)!}}{\frac{N^{N-k}}{(N-k)!}\left(1 + \frac{N-k}{N} + \frac{N-k}{N}\frac{N-k-1}{N} + \dots\right)}$$

$$= \frac{\frac{N^{N-k}}{(N-k)!}}{\frac{N^{N-k}}{(N-k)!}\left(\sum_{n=0}^{\hat{\varepsilon}} \frac{(N-k)!}{N^n(N-k-n)!}\right)} = \frac{1}{\sum_{n=0}^{\hat{\varepsilon}} \frac{(N-k)!}{N^n(N-k-n)!}} < \frac{1}{(\hat{\varepsilon}+1)(1-\frac{1}{\hat{\varepsilon}+1})} = \frac{1}{\hat{\varepsilon}} < \varepsilon.$$

□

We now use Lemmas 2.4 and 2.5 to prove Lemma 2.3(i).

Proof of Lemma 2.3(i). Denote $x = 2\gamma_2\lambda \leq 1$ and $A = \frac{\gamma_2}{\gamma_1} < 1$. Let $\varepsilon > 0$ be given. Fix k^* such that $A^{k^*} < \varepsilon/2$. Lemma 2.5 gives an $\bar{N}$ such that

$$\frac{\frac{(xN)^{N-k}}{(N-k)!}}{\sum_{n=0}^{N} \frac{(xN)^n}{n!}} < \frac{\varepsilon}{2k^*},$$

for all $N > \bar{N}$ and $k = 0, \dots, k^* - 1$ (we can use Lemma 2.5 for all k and choose the maximum over all values of $\bar{N}$). Then, complete the proof by observing that

$$Z(N,\lambda,\gamma_1,\gamma_2) = A\frac{\sum_{n=0}^{N} A^{N-n}\frac{(xN)^n}{n!}}{\sum_{n=0}^{N} \frac{(xN)^n}{n!}} \leq A\left(k^*\frac{\varepsilon}{2k^*} + \frac{\sum_{n=0}^{N-k^*} A^{N-n}\frac{(xN)^n}{n!}}{\sum_{n=0}^{N} \frac{(xN)^n}{n!}}\right)$$

$$\leq \frac{\varepsilon}{2} + A^{k^*}\frac{\sum_{n=0}^{N-k^*} A^{N-n-k^*}\frac{(xN)^n}{n!}}{\sum_{n=0}^{N} \frac{(xN)^n}{n!}} < \frac{\varepsilon}{2} + A^{k^*} < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} < \varepsilon$$

for all $N > \bar{N}$. □

Next assume $x > 1$. We need the following supporting result.

Lemma 2.6. *For* $0 \le A \le 1$ *and* $x > 1$,

$$Y(N) := \frac{\sum_{n=0}^{N} A^{N-n} \frac{(xN)^n}{n!}}{\frac{(xN)^N}{N!}} \overset{N\to\infty}{\to} \frac{1}{1-\frac{A}{x}} \quad \text{for all} \quad 0 \le A \le 1.$$

Proof. For $\varepsilon > 0$, choose k^* such that

$$\left(\frac{A}{x}\right)^{k^*+1} < \left(1 - \frac{A}{x}\right)\frac{\varepsilon}{2}.$$

We use Lemma 2.4 with $k = 0$ to find $\bar{N}$ such that for all $N > \bar{N}$,

$$\frac{N!}{N^n(N-n)!} > 1 - \frac{\varepsilon}{2}\left(1 - \frac{A}{x}\right)$$

for all $n \le k^*$. Then we obtain for all $N > \bar{N}$,

$$\begin{aligned} Y(N) &= \sum_{n=0}^{N}\left(\frac{A}{x}\right)^n \frac{N!}{N^n(N-n)!} \ge \sum_{n=0}^{k^*}\left(\frac{A}{x}\right)^n \left(1 - \frac{\varepsilon}{2}\left(1 - \frac{A}{x}\right)\right) \\ &= \left(1 - \frac{\varepsilon}{2}\left(1 - \frac{A}{x}\right)\right)\frac{1 - \left(\frac{A}{x}\right)^{k^*+1}}{1 - \frac{A}{x}} \\ &= \frac{1}{1-\frac{A}{x}} - \frac{\varepsilon}{2}\frac{1-\frac{A}{x}}{1-\frac{A}{x}} - \frac{\left(\frac{A}{x}\right)^{k^*+1}}{1-\frac{A}{x}} + \frac{\varepsilon}{2}\left(1 - \frac{A}{x}\right)\frac{\left(\frac{A}{x}\right)^{k^*+1}}{1-\frac{A}{x}} > \frac{1}{1-\frac{A}{x}} - \frac{\varepsilon}{2} - \frac{\varepsilon}{2} = \frac{1}{1-\frac{A}{x}} - \varepsilon. \end{aligned}$$

Since for all n,

$$\frac{N!}{N^n(N-n)!} \le 1,$$

this gives

$$\sum_{n=0}^{N}\left(\frac{A}{x}\right)^n \frac{N!}{N^n(N-n)!} \le \sum_{n=0}^{N}\left(\frac{A}{x}\right)^n \to \frac{1}{1-\frac{A}{x}} \text{ as } N \to \infty.$$

We have shown that in the limit

$$\frac{1}{1-\frac{A}{x}} - \varepsilon \le Y(N) \le \frac{1}{1-\frac{A}{x}},$$

which gives the desired result. □

Proof of Lemma 2.3(ii). We apply Lemma 2.6 to both the numerator and denominator of $Z(N,\lambda,\gamma_1,\gamma_2)$ (in the denominator we use $A = 1$) to prove (ii) of Lemma 2.3. □

We will now apply Lemma 2.3 to (2.38) to find

$$\lim_{N\to\infty} B(2,N,\lambda,\alpha_1,\alpha_2,\beta_1,\beta_2) = \begin{cases} 2\beta_1\left(\frac{\alpha_1}{2\beta_1} - \frac{\alpha_2}{2\beta_2}\right) & \text{if } 2\gamma_2\lambda \le 1, \\ 1 - 1/\lambda & \text{if } 2\gamma_2\lambda > 1, \end{cases} \tag{2.40}$$

which after using $\alpha_1 + \alpha_2 = 1$ and $\beta_1 + \beta_2 = 1$ gives Proposition 2.1.

2.B Proof of Proposition 2.2

We first derive the following supporting result.

Proposition 2.3 (Recursive formula for the blocking probability)**.**

$$B(R+1,N,\lambda) = \frac{R}{N+R-\lambda N(1-B(R,N,\lambda))}. \tag{2.41}$$

The proof of Proposition 2.3 is straightforward but mathematically involved.

Proof. We use the Pochhammer symbol $(a)_n = a(a+1)(a+2)\cdot\ldots\cdot(a+n-1) = \frac{(a+n-1)!}{(a-1)!}$ and rewrite the blocking probability as

$$B(R+1,N,\lambda) = R\frac{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N}(N-n+1)_R\frac{(\lambda N)^n}{n!}} = R\frac{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N}(N-n+R)(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}.$$

Then

$$\begin{aligned}\frac{R}{B(R+1,N,\lambda)} &= \frac{\sum_{n=0}^{N}(N-n+R)(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}} \\ &= N+R-\frac{\sum_{n=0}^{N}n(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}.\end{aligned} \tag{2.42}$$

We next rewrite the numerator in (2.42) as

$$\sum_{n=0}^{N} n(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!} = \lambda N\sum_{n=0}^{N}(N-n)_{R-1}\frac{(\lambda N)^n}{n!}. \tag{2.43}$$

Substituting (2.43) in (2.42) yields

$$\begin{aligned}
\frac{R}{B(R+1,N,\lambda)} &= N+R-\lambda N\frac{\sum_{n=0}^{N}(N-n)_{R-1}\frac{(\lambda N)^n}{(n)!}}{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}\\
&= N+R-\lambda N\frac{\sum_{n=0}^{N}(N-n)_{R-1}\frac{(\lambda N)^n}{(n)!}-\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}-\lambda N\\
&= N+R+\lambda N\frac{\sum_{n=0}^{N}(R-1)(N-n+1)_{R-2}\frac{(\lambda N)^n}{(n)!}}{\sum_{n=0}^{N}(N-n+1)_{R-1}\frac{(\lambda N)^n}{n!}}-\lambda N\\
&= N+R-\lambda N(1-B(R,N,\lambda)),
\end{aligned}$$

which gives Proposition 2.3. □

Remark. Equation (2.41) in Proposition 2.3 can be rewritten to

$$\begin{aligned}
&(N+R-1)\frac{{}_1F_1(-N,1-N-R,\lambda N)}{{}_1F_1(-N,2-N-R,\lambda N)}\\
&= N+R-1-\lambda N\left(1-\left(\frac{R-2}{N+R-2}\right)\frac{{}_1F_1(-N,3-N-R,\lambda N)}{{}_1F_1(-N,2-N-R,\lambda N)}\right),
\end{aligned}$$

in which

$${}_1F_1(a,b,z)=\sum_{n=0}^{\infty}\frac{a_{(n)}z^n}{b_{(n)}n!}$$

is Kummer's function. After substituting $a := -N$, $b := 2-N-R$ and $z := \lambda N$, and multiplying with the function in the denominator, one can use (13.3.2) of [DLMF] to obtain Proposition 2.3 [Tem16].

Proof. First assume that $\lambda < 1$. Using Proposition 2.3 and $B(R,N,\lambda)\geq 0$ gives

$$B(R+1,N,\lambda)=\frac{R}{R+N(1-\lambda)+\lambda NB(R,N,\lambda)}\leq\frac{R}{R+N(1-\lambda)}\to 0 \quad \text{as} \quad N\to\infty.$$

For $\lambda = 1$, we have $B(R+1,N,\lambda)=\frac{R}{R+NB(R,N,\lambda)}$. We use mathematical induction to prove that for each R there exist M_1 and M_2 so that $M_1\leq\sqrt{N}B(R,N,1)\leq M_2$ for large enough N. The induction basis follows from [Jag74] since $\sqrt{N}B(1,N,1)\to\sqrt{2/\pi}$. Next, we assume that $M_1\leq\sqrt{N}B(R,N,1)\leq M_2$. Then,

$$\frac{R\sqrt{N}}{R+\sqrt{N}M_2}\leq\sqrt{N}B(R+1,N,1)\leq\frac{R\sqrt{N}}{R+\sqrt{N}M_1},$$

so that the statement also holds for $R+1$ for large enough N. Since $\sqrt{N}B(R,N,1)$ is bounded between two constants, we know that $B(R,N,1) \to 0$ which proves Proposition 2.2. □

Remark. We use Proposition 2.3 to gain some insights in the case $R = 2$. We obtain

$$B(2,N,\lambda) = \frac{R}{N + R - \lambda N(1 - B_e(N,\lambda N))},$$

in which

$$B_e(N,\lambda N) = B(1,N,\lambda) = \frac{\frac{(\lambda N)^N}{N!}}{\sum_{n=0}^{N} \frac{(\lambda N)^n}{n!}}$$

is the Erlang-B formula for the blocking probability in a standard M/M/N/N system with arrival rate λN and unit mean service times. We see that knowledge of the basic Erlang-B formula immediately translates into results for the blocking scenario for $R = 2$. For instance, for $\rho < 1$, $B_e(N,\rho N) \to 0$ as $N \to \infty$ [Jag74], so that for all $\lambda < 1$,

$$B(2,N,\lambda) \approx \frac{1}{1 + N(1-\lambda)} \to 0 \text{ as } N \to \infty.$$

This approximation is good for small λ and large N. When λ is close to 1, the approximation becomes accurate only for really large N. In that case, more precise asymptotic expansions may give better approximations; see [Jag74; JLZ08; LT09]. For example, we can use that $\sqrt{N}B_e(N,N) \to \sqrt{2/\pi}$ to obtain

$$B(2,N,1) \approx \frac{1}{1 + \sqrt{2N/\pi}},$$

which gives $B(2,N,1) \to 0$ and $\sqrt{N}B(2,N,1) \to \sqrt{\pi/2}$ as $N \to \infty$.

Chapter 3

Hyper-scalable load balancing with scheduled updates

Based on:

[BBL19] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. "Hyper-Scalable JSQ with Sparse Feedback". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3.1 (2019)

3.1 Introduction

We introduce and analyze a novel class of hyper-scalable LBAs (load balancing algorithms) that leverage memory at the (single) dispatcher, in order to counter the scalability challenges as mentioned in Section 1.2. The basic scheme is as follows:

Algorithm 3.1 (Basic hyper-scalable scheme)**.** *The dispatcher forwards incoming jobs to the server with the lowest queue estimate. The dispatcher maintains an estimate for every server and increments these estimates for every job that is assigned. Status updates of servers occur at rate δ per server, and update the estimate that the dispatcher has at its disposal to the actual queue length.*

We introduce four hyper-scalable schemes that obey the rules of Algorithm 3.1 but differ in when the status updates are sent. When the updates sent by the servers to the dispatcher are synchronized, we denote the scheme by SUJSQ(δ) (Synchronized-Updates JSQ). Similarly we introduce AUJSQ(δ) (Asynchronous-Updates JSQ), which is used when the updates are asynchronous. We add an exp-tag whenever the time between two updates is exponentially distributed (with parameter δ and mean $1/\delta$) and a det-tag when the time between the updates is constant ($1/\delta$). This gives rise to four schemes; SUJSQ$^{\text{det}}(\delta)$, SUJSQ$^{\text{exp}}(\delta)$, AUJSQ$^{\text{det}}(\delta)$ and AUJSQ$^{\text{exp}}(\delta)$.

We show that the four schemes can achieve a vanishing waiting time in the many-server limit with just one message per job if the load is low enough. The proposed schemes are particularly geared however towards the sparse feedback regime with less than one message per job, where they outperform corresponding sparsified JIQ versions. With fluid limits (see Section 1.3.2) we demonstrate that in the ultra-low feedback regime the mean stationary waiting time tends to a constant in the synchronized case, but grows without bound in the asynchronous case. A more detailed overview of our key findings is presented in the next section.

Organization of the chapter. In Section 3.2 we discuss our key findings and contributions for the hyper-scalable schemes, obtained through fluid-limit analysis and extensive simulations, and we introduce some useful notation and preliminaries. We turn to a comprehensive analysis of the synchronized and asynchronous cases through the lens of fluid limits in Sections 3.3 and 3.4, respectively. In Section 3.5 we conclude with some summarizing remarks, a comparison with the related work [AD20] and topics for further research.

3.2 Model description and key results

The precise model we consider consists of N parallel identical servers and one dispatcher. Jobs arrive at the dispatcher as a Poisson process of rate λN, where λ denotes the job arrival rate per server. Every job is dispatched to one of the servers, after which it joins the queue of the server if the server is busy, or will start its service when the server is idle. The job processing requirements are independent and exponentially distributed with unit mean at each of the servers. We consider several load balancing algorithms for the dispatching of jobs to servers, including the hyper-scalable schemes SUJSQ$^{\text{det}}(\delta)$, SUJSQ$^{\text{exp}}(\delta)$, AUJSQ$^{\text{det}}(\delta)$ and AUJSQ$^{\text{exp}}(\delta)$. In the simulation experiments we will also briefly consider SUJSQ$^{\text{det,idle}}(\delta)$, which is similar to SUJSQ$^{\text{det}}(\delta)$, except that only idle servers send notifications. We now present the results from simulation studies and the fluid-limit and fixed-point analysis in Sections 3.3 and 3.4.

3.2.1 Large-system performance

We investigate fluid limits in order to explore the performance of the hyper-scalable algorithms in the many-server limit $N \to \infty$. We analyze their behavior and fixed points, and use these to derive results for the system in stationarity as function of the update frequency δ.

Asymptotically optimal feedback regime. Using fluid-limit analysis, we prove that the proposed schemes can achieve a vanishing waiting time in the many-server limit when the update frequency δ exceeds $\lambda/(1-\lambda)$. In case servers only report zero queue lengths and suppress updates for non-zero queues, the update frequency required for a vanishing waiting time can in fact be lowered to just λ, matching the one message per job involved in the JIQ scheme.

Sparse feedback regime. Figure 3.1 displays results from extensive simulations and shows the mean waiting time as function of the number of messages per job. This number is proportional to the update frequency δ, and equals δ/λ for the four hyper-scalable schemes. We also show results for JIQ(p), a sparsified version of JIQ, where a token is sent to the dispatcher with probability p whenever a server becomes idle. Random refers to the scheme where every job is assigned to a server selected uniformly at random, and Round-Robin assigns the i-th arriving job to server $1 + i \mod N$.

Figure 3.1 further shows that the schemes SUJSQ$^{\text{det}}(\delta)$ and SUJSQ$^{\text{exp}}(\delta)$ outperform JIQ(p) in the sparse feedback regime when $\delta < 0.5$. Also observe that SUJSQ$^{\text{det,idle}}(\delta)$, the scheme in which only idle servers send reports, achieves a near-zero waiting time with just one message per job, just like the JIQ scheme, and outperforms JIQ(p) across most of the relevant domain $\delta < 0.5$. However, as $\delta \downarrow 0$ the waiting time grows without bound, since estimates will grow large due to lack of updates, which causes servers that are reported idle in the latest update to receive many jobs in succession.

Ultra-low feedback regime. We examine the performance in the ultra-low feedback regime where the update frequency δ goes to zero, and in particular establish a somewhat counter-intuitive dichotomy. When all status updates are synchronized, the behavior of each of the individual queues approaches that of a single-server queue with a near-deterministic arrival process and exponential service times, with the mean stationary waiting time tending to a finite constant. In contrast, for asynchronous updates, the individual queues experience saw-tooth behavior with oscillations and waiting times that grow without bound.

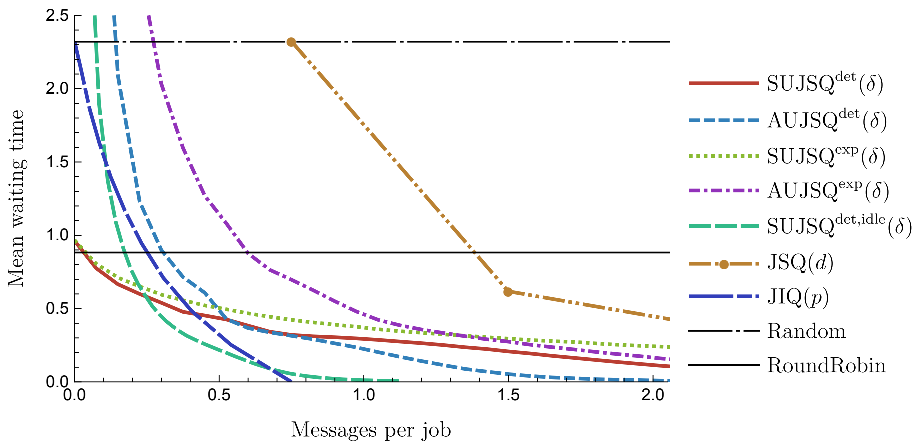


Figure 3.1: Mean waiting times for different schemes (with various parameters) for $\lambda = 0.7$ and $N = 200$ obtained via a discrete-event simulation. For each of the schemes, the parameter δ, d or p is varied and the communication in terms of messages per job and the waiting times of jobs are tracked in the simulation.

3.2.2 Synchronize or not?

In case of synchronized updates, the dispatcher will update the queue lengths of the servers simultaneously. Thus, just after an update moment, the dispatcher has a perfect view of the status of all servers and it will dispatch jobs optimally. After a while, the estimates will start to deviate from the actual queue lengths, so that the scheme no longer makes (close to) optimal decisions. With asynchronous updates servers send updates at independent times, which means that some of the estimates may be very accurate, while others may differ significantly from the actual queue lengths.

Round-Robin resemblance. We find that both $\mathrm{SUJSQ}^{\mathrm{det}}(\delta)$ and $\mathrm{SUJSQ}^{\mathrm{exp}}(\delta)$ resemble Round-Robin as the update frequency δ approaches zero, and are the clear winners in the ultra-low feedback regime, which is crucial from a scalability perspective (see Figure 3.1). To understand the resemblance with Round-Robin, notice that the initial queue lengths after an update will be small compared to the number of arrivals until the next update. Thus soon after the update the dispatcher will essentially start forwarding jobs in a (probabilistic) Round-Robin manner. Specifically, most servers will have equal queue estimates at certain points in time, and they will each receive one job every λ time units, but in a random order. This pattern repeats itself and resembles Round-Robin, where the difference of received jobs among servers can be at most one.

Dichotomy. In Figure 3.1 we also see that while AUJSQ$^{\text{det}}(\delta)$ outperforms the synchronized variants for large values of the update frequency δ, it produces a mean waiting time that grows without bound as δ approaches zero. The latter issue also occurs for AUJSQ$^{\text{exp}}(\delta)$ and render the asynchronous versions far inferior in the ultra-low feedback regime compared to both synchronized variants. To understand this remarkable dichotomy, notice that queue estimates must inevitably grow to increasingly large values of the order λ/δ and significantly diverge from the true queue lengths as the update frequency becomes small, both in the synchronized and asynchronous versions. However, in the synchronized variants the queue estimates will all be lowered and updated to the true queue lengths simultaneously, prompting the dispatcher to evenly distribute incoming jobs over time. In contrast, in the asynchronous versions, a server will be the only one with a low queue estimate right after an update, and almost immediately be assigned a huge pile of jobs to bring its queue estimate at par, resulting in oscillatory effects. This somewhat counter-intuitive dichotomy reveals that the synchronized variants behave benignly in the presence of outdated information, while the asynchronous versions are adversely impacted.

3.2.3 Notation and preliminaries

In this subsection we introduce some useful notation and preliminaries in preparation for the fluid-limit analysis in Sections 3.3 and 3.4. Recall that all the servers are identical and the dispatcher only distinguishes among servers based on their queue estimates and does not take their identities into account when forwarding jobs. Hence we do not need to keep track of the state of each individual server, but only count the number of servers that reside in a particular state. Specifically, we will denote by $Y_{i,j}(t)$ the number of servers with queue length i (including a possible job being served) and queue estimate $j \geq i$ at the dispatcher at time t. Further denote by $V_i = \sum_{l=i}^{\infty} Y_{il}$ and $W_j = \sum_{k=0}^{j} Y_{kj}$ the total number of servers with queue length i and queue estimate j, respectively, when the system is in state Y.

In order to analyze fluid limits in a regime where the number of servers N grows large, we will consider a sequence of systems indexed by N, and attach a superscript N to the associated state variables. We specifically introduce the fluid-scaled state variables $y_{i,j}^N(t) = Y_{i,j}^N(t)/N$, representing the fraction of servers in the N-th system with true queue length i and queue estimate $j \geq i$ at the dispatcher at time t, and assume that the sequence of initial states is such that $y^N(0) \to y^\infty$. Any (weak) limit $\{y(t)\}_{t\geq 0}$ of the sequence $(\{y^N(t)\}_{t\geq 0})_{N\geq 1}$ as $N \to \infty$ will be called a fluid limit (also see Section 1.3.2).

Let $m(Y) = \min\{j : W_j > 0\}$ be the minimum queue estimate among all servers when the system is in state Y. When a job arrives and the system is in state Y, it is dispatched to one of the $W_{m(Y)}$ servers with queue estimate $m(Y)$ selected uniformly at random, so it joins a server with queue length $i \leq m(Y)$ with probability $Y_{i,m(Y)}/W_{m(Y)}$. Because of the Poisson arrival process, transitions from a state Y to a state Y' with $Y'_{i,j} = Y_{i,j} - 1$ and $Y'_{i+1,j+1} = Y_{i+1,j+1} + 1$ thus occur at rate $\lambda N p_{i,j}(Y)$, with $p_{i,j}(Y) = \mathbb{1}\{j = m(Y)\} Y_{i,j}/W_j$, $i = 0, \ldots, j$. Due to the unit-exponential processing requirements, transitions from a state Y to a state Y' with $Y'_{i,j} = Y_{i,j} - 1$ and $Y'_{i-1,j} = Y_{i-1,j} + 1$ occur at rate $Y_{i,j}$, $i = 1, \ldots, j$. For notational compactness, we further omit the dependence of Y for $m(Y)$, $p_{i,j}(Y)$ and $q_{i,j}(Y)$ and instead write $m(t)$, $p_{i,j}(t)$ and $q_{i,j}(t)$ as they only depend on t though $y(t)$.

In order to specify the transitions due to the updates, we need to distinguish between the synchronized and the asynchronous case.

Synchronized updates

Whenever a synchronized update occurs and the system is in state Y, a transition occurs to state Y' with $Y'_{ii} = V_i$ and $Y'_{ij} = 0$ for $i = 0, \ldots, j-1$. Note that these transitions only occur in a Markovian fashion when the update intervals are exponentially distributed. When the update intervals are non-exponentially distributed, $\{Y(t)\}_{t\geq 0}$ is not a Markov process, but the evolution between successive updates is still Markovian.

Asynchronous updates

When the system is in state Y and a server with queue length i and queue estimate $j > i$ sends an update to the dispatcher, a transition occurs to a state Y' with $Y'_{i,j} = Y_{i,j} - 1$ and $Y'_{i,i} = Y_{i,i} + 1$. Note that these transitions only occur in a Markovian fashion when the update intervals are exponentially distributed. When the update intervals are non-exponentially distributed, $\{Y(t)\}_{t\geq 0}$ is not a Markov process, and in order to obtain a Markovian state description, the state variables Y_{ij} would in fact need to be augmented with continuous variables keeping track of the most recent update moments for the various servers.

3.3 Synchronized updates

In this section we examine the fluid limit for synchronized updates. In Section 3.3.1 we provide a description of the fluid-limit trajectory, along with an intuitive interpretation, numerical illustration and comparison with simulation. In Section 3.3.2 the fluid-limit analysis will be used to derive a finite upper bound for the queue length on fluid scale for any given update frequency $\delta > 0$ (Proposition 3.1) and to show that in the long term queueing vanishes on fluid level for a sufficiently high update frequency δ (Proposition 3.2).

3.3.1 Fluid-limit dynamics

The fluid limit $y(t)$ (in between successive update moments) satisfies the system of differential equations

$$\frac{\mathrm{d}y_{i,j}(t)}{\mathrm{d}t} = y_{i+1,j}(t)\mathbb{1}\{i < j\} - y_{i,j}(t)\mathbb{1}\{i > 0\} + \lambda p_{i-1,j-1}(t)\mathbb{1}\{i > 0\} - \lambda p_{i,j}(t), \quad (3.1)$$

where

$$p_{i,j}(t) = \frac{y_{i,j}(t)}{w_j(t)}\mathbb{1}\{m(t) = j\}$$

denotes the fraction of jobs assigned to a server with queue length i and queue estimate j in fluid state y at time t, with $w_j(t) = \sum_{k=0}^{j} y_{k,j}(t)$ denoting the fraction of servers with queue estimate j and $m(t) = \min(j | w_j(t) > 0)$ the minimum queue estimate in fluid state y at time t. At an update moment τ, the fluid limit shows discontinuous behavior, with $y_{i,i}(\tau) = v_i(\tau^-)$ and $y_{i,j}(\tau) = 0$ for all $i < j$, with $v_i(t) = \sum_{l=i}^{\infty} y_{i,l}(t)$ denoting the fraction of servers with queue length i in fluid state y at time t.

Informal outline of the derivation

We now provide an informal outline of the derivation of the fluid limit for synchronized updates as stated in (3.1). Let $A_{i,j}(t)$ and $S_{i,j}(t)$ denote unit-rate Poisson processes, $j \geq i \geq 0$, all independent.

The system dynamics (in between successive update moments) may then be represented as (see for instance [HK94; PTW07])

$$Y_{i,j}^N(t) = Y_{i,j}^N(t_0) + S_{i+1,j}\left(\int_{t_0}^t Y_{i+1,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i<j\} - S_{i,j}\left(\int_{t_0}^t Y_{i,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i>0\}$$
$$+ A_{i-1,j-1}\left(\lambda N\int_{t_0}^t p_{i-1,j-1}(Y^N(s))\mathrm{d}s\right)\mathbb{1}\{i>0\} - A_{i,j}\left(\lambda N\int_{t_0}^t p_{i,j}(Y^N(s))\mathrm{d}s\right),$$

with $p_{i,j}(Y) = \frac{Y_{i,j}}{\sum_{k=0}^j Y_{k,j}}\mathbb{1}\{j = m(Y)\}$.

Dividing by N and rewriting in terms of the fluid-scaled variables $y_{i,j}^N(t) = \frac{1}{N}Y_{i,j}^N(t)$, we obtain

$$\begin{aligned} y_{i,j}^N(t) &= y_{i,j}^N(t_0) \\ &\quad + \frac{1}{N}S_{i+1,j}\left(N\int_{t_0}^t y_{i+1,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i<j\} - \frac{1}{N}S_{i,j}\left(N\int_{t_0}^t y_{i,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i>0\} \\ &\quad + \frac{1}{N}A_{i-1,j-1}\left(\lambda N\int_{t_0}^t p_{i-1,j-1}(Y^N(s))\mathrm{d}s\right)\mathbb{1}\{i>0\} \\ &\quad - \frac{1}{N}A_{i,j}\left(\lambda N\int_{t_0}^t p_{i,j}(Y^N(s))\mathrm{d}s\right). \end{aligned} \tag{3.2}$$

Now introduce

$$\tilde{S}_{k,l}(u) := S_{k,l}(u) - u, \quad \tilde{A}_{k,l}(u) := A_{k,l}(u) - u,$$

and observe that $\tilde{S}_{k,l}(\cdot)$ and $\tilde{A}_{k,l}(\cdot)$ are martingales. By standard arguments it can be shown that both $\frac{1}{N}\tilde{S}_{k,l}(N\int_{t_0}^t y_{k,l}^N(s)\mathrm{d}s)$ and $\frac{1}{N}\tilde{A}_{k,l}(\lambda N\int_{t_0}^t p_{k,l}(Y^N(s))\mathrm{d}s)$ converge to zero as $N\to\infty$ almost surely.

Exploiting the fact that the minimum queue estimate $m(Y^N(t))$ cannot decrease in between successive update moments, it can also be established that

$$\int_{t_0}^t p_{k,l}(Y^N(s))\mathrm{d}s \to \int_0^t p_{k,l}(s)\mathrm{d}s,$$

as $N\to\infty$, with $p_{k,l}(y) = \frac{y_{k,l}}{w_l(y)}\mathbb{1}\{m(y) = l\}$ as defined earlier.

Taking the limit for $N\to\infty$ in (3.2), we conclude that any (weak) limit $\{y_{i,j}(t)\}_{t\geq 0}$ of the sequence $\left(\{y_{i,j}^N(t)\}_{t\geq 0}\right)_{N\geq 1}$ in between successive update moments must satisfy

$$y_{i,j}(t) = y_{i,j}(t_0) + \int_{t_0}^{t} y_{i+1,j}(s)\mathrm{d}s\mathbb{1}\{i < j\} - \int_{t_0}^{t} y_{i,j}(s)\mathrm{d}s\mathbb{1}\{i > 0\}$$
$$+ \lambda \int_{t_0}^{t} p_{i-1,j-1}(y(s))\mathrm{d}s\mathbb{1}\{i > 0\} - \lambda \int_{t_0}^{t} p_{i,j}(y(s))\mathrm{d}s,$$

with $y_{i,j}(0) = y_{i,j}^{\infty}$.

Rewriting the latter integral equation in differential form yields (3.1).

Interpretation

The above system of differential equations may be heuristically explained as follows. The first two terms correspond to service completions at servers with $i+1$ and i jobs, which result in an increase and decrease in the fraction of servers with i jobs, respectively. The third and fourth terms reflect the job assignments to servers with the minimum queue estimate $m(t)$. The third term captures the resulting increase in the fraction of servers with queue estimate $m(t)+1$, while the fourth term captures the corresponding decrease in the fraction of servers with queue estimate $m(t)$.

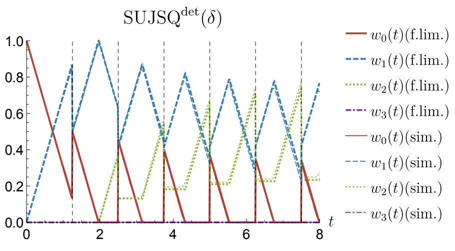
Summing the equations (3.1) over $i = 0, 1, \ldots, j$ yields

$$\frac{\mathrm{d}w_j(t)}{\mathrm{d}t} = \lambda[\mathbb{1}\{m(t) = j-1\} - \mathbb{1}\{m(t) = j\}] = \begin{cases} -\lambda & j = m(t), \\ \lambda & j = m(t)+1, \\ 0 & j \neq m(t), m(t)+1, \end{cases} \tag{3.3}$$

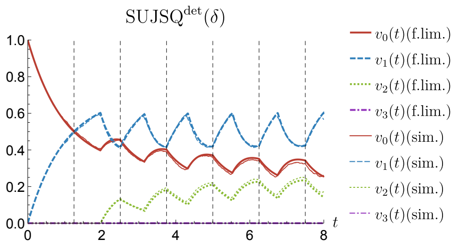
reflecting that servers with the minimum queue estimate $m(t)$ are assigned jobs, and flipped into servers with queue estimate $m(t)+1$, at rate λ, and that $m(t)$ can only increase between successive update moments. Also note that the derivative of $y_{i,j}$ is continuous in t, except at those times t where $m(t)$ increases, and that $y_{i,j}(t)$ is continuous in between updates since $\mathrm{d}y_{i,j}(t)/\mathrm{d}t$ is bounded.

Numerical illustration and comparison with simulation

Figures 3.2a, 3.2b, 3.3a and 3.3b show the fluid-limit trajectories $y(t)$ as governed by the differential equations in (3.1) for SUJSQ$^{\text{det}}(\delta)$, along with (fluid-scaled) variables $y_{i,j}^{N}(t)$ obtained through stochastic simulation for a system with $N = 1000$ servers and averaged over 10 runs. Observe that the simulation results are nearly indistinguishable from the fluid-limit trajectories, which is in

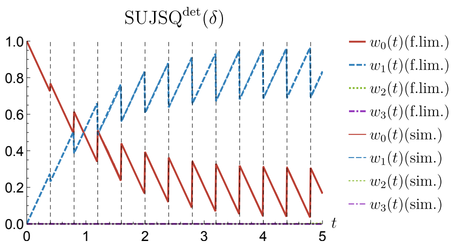


(a)

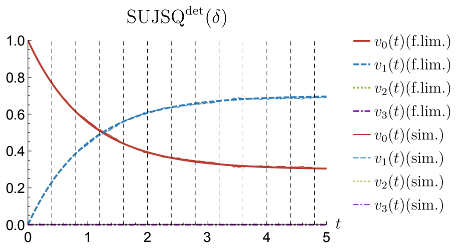


(b)

Figure 3.2: Numerical emulation of the fluid limit for $\mathrm{SUJSQ}^{\mathrm{det}}(0.85)$ and $\lambda = 0.7$, accompanied by simulation results for $N = 1000$, averaged over 10 runs.



(a)



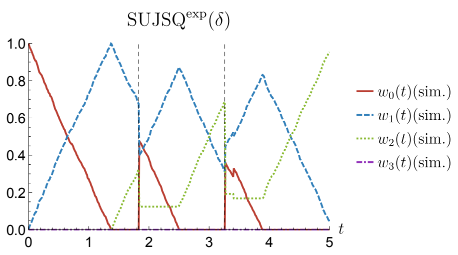
(b)

Figure 3.3: Numerical emulation of the fluid limit for $\mathrm{SUJSQ}^{\mathrm{det}}(2.5)$ and $\lambda = 0.7$, accompanied by simulation results for $N = 1000$, averaged over 10 runs.

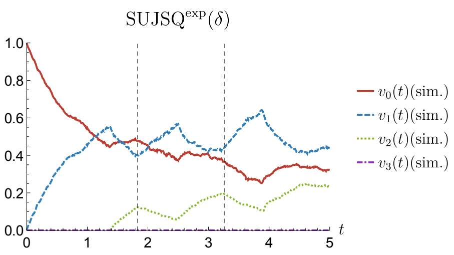
line with broader findings concerning the accuracy of fluid and mean-field limits [Gas17; Yin16].

Update moments at times $1/\delta, 2/\delta, \ldots$ are marked by vertical dotted lines. These time points can also be easily recognized by the jumps in the fraction of servers $w_j(t)$ that have queue estimate j. Moreover, the fraction of servers $v_i(t)$ with queue length i is not differentiable in these points as well as other moments when the minimum queue estimate changes.

In Figures 3.2a and 3.2b, $\delta = 0.85 < \lambda/(1-\lambda) = 7/3$, while $\delta = 2.5 > 7/3$ in Figures 3.3a and 3.3b. In the first scenario there are moments at which $w_0(t)$ is zero and some jobs are sent to servers with one job in queue already. This results in servers sometimes having two jobs, which means that queueing does not vanish as $N \to \infty$. In contrast, in the second scenario, $w_0(t)$ is strictly positive

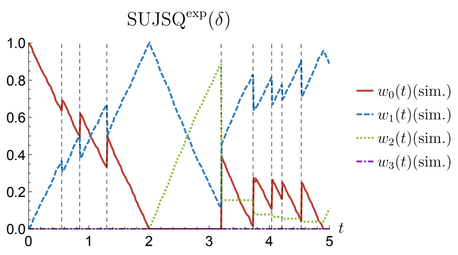


(a)

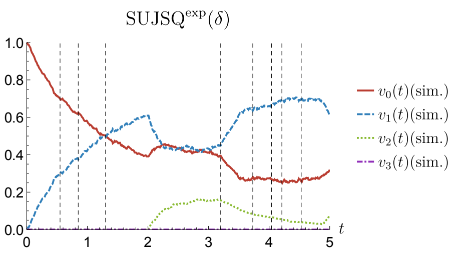


(b)

Figure 3.4: Simulation results for $\mathrm{SUJSQ}^{\mathrm{exp}}(0.85)$.



(a)



(b)

Figure 3.5: Simulation results for $\mathrm{SUJSQ}^{\mathrm{exp}}(2.5)$.

at all times and no servers appear with two or more jobs, which implies that no queueing occurs at fluid level as $N \to \infty$. We will return to this dichotomy in Proposition 3.2.

Qualitatively similar results are observed for $\mathrm{SUJSQ}^{\mathrm{exp}}(\delta)$, where the updates occur at irregular moments. The paths still follow similar saw-tooth patterns, and the dynamics between updates are identical, as reflected in the differential equations in (3.1). In particular, the fraction of servers with minimum queue estimate decreases linearly between updates, and the estimate drastically changes at update moments. The results are displayed in Figures 3.4a, 3.4b, 3.5a and 3.5b, for a system with $N = 1000$ servers and $\lambda = 0.7$. Note that a large value of δ does not guarantee vanishing queueing anymore as there is a positive probability that the time between two updates is too long.

3.3.2 Performance in the fluid limit

We will now use the fluid limit (3.1) to gain some insight in the performance of the SUJSQ$^{\text{det}}(\delta)$ scheme. Equation (3.3) shows that no fixed point can exist as $w_j(t)$ has a non-zero constant derivative. We will establish however in Proposition 3.1 that for any positive update frequency the queue lengths on fluid scale are essentially bounded by a finite constant. Furthermore, in Proposition 3.2 we demonstrate that when the update frequency is above a specific threshold, queueing basically vanishes on fluid level in the long term.

Consider the average queue length on fluid scale, denoted and defined by

$$Q(t) = \sum_{i=0}^{\infty} i v_i(t) = \sum_{i=0}^{\infty} i \sum_{j=i}^{\infty} y_{i,j}(t).$$

Further define $z_k(t) = \sum_{i=k}^{\infty} v_i(t)$ as the fraction of servers with queue length k or larger at time t on fluid scale, and note that $Q(t) = \sum_{k=1}^{\infty} z_k(t)$. We will also refer to $Q(t)$ as the total queue 'mass' on fluid scale at time t, and introduce

$$\begin{aligned} Q^{\leq K}(t) &= \sum_{k=1}^{K} z_k(t) = \sum_{k=1}^{K} \sum_{i=k}^{\infty} v_i(t) = \sum_{i=1}^{K} \sum_{k=1}^{i} v_i(t) + \sum_{i=K+1}^{\infty} \sum_{k=1}^{K} v_i(t) \\ &= \sum_{i=1}^{K} i v_i(t) + \sum_{i=K+1}^{\infty} K v_i(t) = \sum_{i=1}^{\infty} \min\{i, K\} v_i(t), \end{aligned}$$

and

$$\begin{aligned} Q^{>K}(t) = Q(t) - Q^{\leq K}(t) &= \sum_{k=K+1}^{\infty} z_k(t) = \sum_{k=K+1}^{\infty} \sum_{i=k}^{\infty} v_i(t) \\ &= \sum_{i=K+1}^{\infty} \sum_{k=K+1}^{i} v_i(t) = \sum_{i=K+1}^{\infty} (i-K) v_i(t) \end{aligned}$$

as the queue mass (weakly) below and (strictly) above level K, respectively.

We first focus on the evolution of the fluid limit in between updates and consider t to lie inside the interval $[0, \tau]$ with $\tau = 1/\delta$. The fluid-limit equation (3.1) yields the following expression for the derivative of $z_k(t)$,

$$\begin{aligned}
\frac{\mathrm{d}z_k(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=k}^{\infty} v_i(t) = \frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=k}^{\infty}\sum_{j=i}^{\infty} y_{i,j}(t) = \sum_{i=k}^{\infty}\sum_{j=i}^{\infty}\frac{dy_{i,j}(t)}{dt} \\
&= \sum_{i=k}^{\infty}\sum_{j=i}^{\infty}\left[y_{i+1,j}(t)\mathbb{1}\{i<j\} - y_{i,j}(t)\mathbb{1}\{i>0\} + \lambda p_{i-1,j-1}(t)\mathbb{1}\{i>0\} - \lambda p_{i,j}(t)\right] \\
&= \sum_{i=k}^{\infty}\left[v_{i+1}(t) - v_i(t)\mathbb{1}\{i>0\} + \lambda\sum_{j=i}^{\infty} p_{i-1,j-1}(t)\mathbb{1}\{i>0\} - \lambda\sum_{j=i}^{\infty} p_{i,j}(t)\right] \\
&= -v_k(t)\mathbb{1}\{k>0\} + \lambda\sum_{j=k-1}^{\infty} p_{k-1,j}(t)\mathbb{1}\{k>0\},
\end{aligned}$$

and

$$\begin{aligned}
\frac{\mathrm{d}Q^{>K}(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\sum_{k=K+1}^{\infty} z_k(t) = \sum_{k=K+1}^{\infty}\frac{\mathrm{d}z_k(t)}{\mathrm{d}t} \\
&= -\sum_{k=K+1}^{\infty} v_k(t)\mathbb{1}\{k>0\} + \lambda\sum_{k=K+1}^{\infty}\sum_{j=k-1}^{\infty} p_{k-1,j}(t)\mathbb{1}\{k>0\} \qquad (3.4) \\
&= -z_{K+1}(t) + \lambda\sum_{k=K}^{\infty}\sum_{j=k}^{\infty}\frac{y_{k,j}(t)}{w_j(t)}\mathbb{1}\{m(t)=j\}.
\end{aligned}$$

This may be interpreted by noting that the queue mass above level K increases due to arriving jobs being assigned to servers with queue length K or larger and decreases due to jobs being completed by servers with queue length $K+1$ or larger. In particular, we find that

$$\frac{\mathrm{d}Q^{>K}(t)}{\mathrm{d}t} = -z_{K+1}(t) \qquad (3.5)$$

for all $K \geq m(t)+1$.

Taking $K=0$ and noting that $Q^{>0}(t) \equiv Q(t)$, we obtain

$$\frac{\mathrm{d}Q(t)}{\mathrm{d}t} = \lambda - z_1(t) = \lambda - [1 - v_0(t)],$$

and thus

$$Q(t) = Q(0) + \lambda t - \int_0^t [1 - v_0(s)]\mathrm{d}s. \qquad (3.6)$$

This reflects that the average queue length on fluid scale at time t is obtained by adding the number of arrivals λt during $[0,t]$ and subtracting the number of service completions, which corresponds to the cumulative fraction of busy servers $1-v_0(s)$.

The next lemma follows directly from (3.5), and shows that the queue mass above level K decreases when the minimum queue estimate on fluid scale is strictly below K, so that there are no arrivals to servers with queue length K or larger.

Lemma 3.1. *If $m(s) \leq K-1$ for all $s \in [0,t)$, then $Q^{>K}(t) \leq Q^{>K}(0)$.*

We now proceed to derive a specific characterization of the decline in the queue mass above level K when the minimum queue estimate on fluid scale is strictly below K.

For conciseness, denote

$$A(L,t) = \sum_{l=0}^{L}(L-l)\alpha_l(t)$$

and

$$B(L,t) := \sum_{l=0}^{L} l\alpha_l(t) + L\sum_{l=L+1}^{\infty}\alpha_l(t)$$

with $\alpha_l(t) = \frac{t^l}{l!}e^{-t}$, and let G be a Poisson random variable with parameter t. Note that $A(L,t) = \mathbb{E}[\max\{L-G,0\}]$ and $B(L,t) = \mathbb{E}[\min\{G,L\}]$ so that $A(L,t) + B(L,t) = L$, and in particular $B(L,t) \leq L$. Observe that $A(L,t)$ may be interpreted as the expected queue length after a time interval of length t at a single server with initial queue length L, unit-exponential service times and no arrivals, while $B(L,t)$ may be interpreted as the expected number of service completions during that time period.

Lemma 3.2. *For any $K \geq 0$, $L \geq 1$, $t \geq 0$,*

$$\int_0^t z_{K+1}(s)\mathrm{d}s \geq \left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right]\left[1-\frac{A(L,t)}{L}\right] = \frac{1}{L}\left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right]B(L,t).$$

The proof of Lemma 3.2 involves a detailed analysis of $z_{K+1}(t)$. In the lemma special attention goes to the mass of jobs that are queued in positions $K+1$ up to $K+L$, represented by $Q^{\leq K+L}(0) - Q^{\leq K}(0)$. The decline in this mass is no less than the decline in a situation where the same total number of jobs reside with servers that have either 0 jobs or exactly L jobs (so a fraction $[Q^{\leq K+L}(0) - Q^{\leq K}(0)]/L$ of the servers will have L jobs). Finally, $A(L,t)$ represents the expected number of jobs that remain at time t at each of the servers with L jobs, while $B(L,t)$ represents the expected number of jobs that have been completed at time t by each of these servers. These observations will be made rigorous in the proof in Section 3.A.1.

The next lemma follows directly from (3.4) and (3.5) in conjunction with Lemma 3.2.

Lemma 3.3. *For any* $K \geq 0,\ L \geq 1,\ t \geq 0,$

$$Q^{>K}(t) \leq \lambda t + Q^{>K+L}(0) + \frac{1}{L}\left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right] A(L,t),$$

or equivalently,

$$Q^{>K}(t) - Q^{>K}(0) \leq \lambda t - \frac{1}{L}\left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right] B(L,t).$$

If $m(y(s)) \leq K-1$ *for all* $s \in [0,t)$*, then for any* $L \geq 1$

$$Q^{>K}(t) \leq Q^{>K+L}(0) + \frac{1}{L}\left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right] A(L,t),$$

or equivalently,

$$Q^{>K}(t) - Q^{>K}(0) \leq -\frac{1}{L}\left[Q^{\leq K+L}(0) - Q^{\leq K}(0)\right] B(L,t).$$

In particular, taking $L = 1$*,*

$$Q^{>K}(t) \leq Q^{>K+1}(0) + \left[Q^{\leq K+1}(0) - Q^{\leq K}(0)\right] \mathrm{e}^{-t} = Q^{>K+1}(0) + z_{K+1}(0)\mathrm{e}^{-t},$$

yielding

$$Q^{>K}(t) \leq \mathrm{e}^{-t} Q^{>K}(0) + [1-\mathrm{e}^{-t}] Q^{>K+1}(0). \tag{3.7}$$

The next lemma provides a simple condition for the minimum queue estimate on fluid scale to remain strictly below K throughout the interval $[0,t)$ in terms of $Q(0)$.

Lemma 3.4. *If* $Q(0) \leq K - \lambda t$*, then* $m(s) \leq K-1$ *for all* $s \in [0,t)$*.*

Proof. We have $m(t) = 0$, $\frac{\mathrm{d}}{\mathrm{d}t} w_0(t) = -\frac{\mathrm{d}}{\mathrm{d}t} w_1(t) = -\lambda$ for $t \in \frac{1}{\lambda}[0, v_0(0)]$, $m(t) = 1$, $\frac{\mathrm{d}}{\mathrm{d}t} w_1(t) = -\frac{\mathrm{d}}{\mathrm{d}t} w_2(t) = -\lambda$ for $t \in \frac{1}{\lambda}[v_0(0), 2v_0(0) + v_1(0)]$, $m(t) = 2$, $\frac{\mathrm{d}}{\mathrm{d}t} w_2(t) = -\frac{\mathrm{d}}{\mathrm{d}t} w_3(t) = -\lambda$ for $t \in \frac{1}{\lambda}[2v_0(0) + v_1(0), 3v_0(0) + 2v_1(0) + v_2(0)]$, ..., $m(t) = j$, $\frac{\mathrm{d}}{\mathrm{d}t} w_j(t) = -\frac{\mathrm{d}}{\mathrm{d}t} w_{j+1}(t) = -\lambda$ for $t \in \frac{1}{\lambda}\left[\sum_{i=0}^{j-1}(j-i)v_i(0),\ \sum_{i=0}^{j}(j+1-i)v_i(0)\right]$, assuming $\frac{1}{\lambda}\sum_{i=0}^{j}(j+1-i)v_i(0) \le t$. In particular $m(s) \le K-1$ for all $s \in [0,t)$ if $\lambda t \le \sum_{i=0}^{K-1}(K-i)v_i(0)$. The latter inequality holds since

$$\begin{aligned}\sum_{i=0}^{K-1}(K-i)v_i(0) &= K\sum_{i=0}^{K-1} v_i(0) - \sum_{i=0}^{K-1} i v_i(0) = K\left(1 - \sum_{i=K}^{\infty} v_i(0)\right) - \sum_{i=0}^{K-1} i v_i(0)\\ &= K - Q^{\le K}(0) \ge K - Q(0) \ge \lambda t.\end{aligned}$$

□

We will henceforth say that L is large enough if

$$\lambda\tau < \sigma(L;\lambda,\tau) = \left(1 - \frac{\lambda\tau+1}{L}\right)B(L,\tau), \tag{3.8}$$

and define $s(\lambda,\tau) = \min\{L : \lambda\tau < \sigma(L;\lambda,\tau)\}$, which may be loosely thought of as the maximum queue length on fluid scale in the sense of the next proposition. Note that $\sigma(L;\lambda,\tau) \uparrow \tau$ as $L \to \infty$, ensuring that $s(\lambda,\tau)$ is finite for any $\lambda < 1$.

Proposition 3.1 (Bounded queue length for SUJSQ$^{\text{det}}(\delta)$). *For any initial state $y(0)$ with finite queue mass $Q(0) < \infty$, the fraction of servers on fluid scale with a queue length larger than $s(\lambda,\tau)$ vanishes over time. Additionally, if the initial queue mass $Q(0)$ is sufficiently small and the initial fraction of servers with a queue length larger than $s(\lambda,\tau)$ is zero, then that fraction will remain zero forever.*

The proof of Proposition 3.1 leverages Lemma 3.2 and is organized as follows. For any initial state, we can show that either the mass in the tail, or the total mass is decreasing. Once one of the two is below a certain level, we show that the other decreases as well. We show this in two lemmas. From that point on, it is a back and forth between decreasing mass in the tail and decreasing total mass, which is described in the final lemma. The mass in the tail will decrease, such that the mass strictly above level $L^* = s(\lambda,\tau)$ will vanish.

Define

$$\Delta = \left(1 - \frac{\lambda\tau+1}{L^*}\right)B(L^*,\tau) - \lambda\tau. \tag{3.9}$$

We now state two corollaries, which provide upper bounds for the total queue mass at time τ. The proofs are based on Lemmas 3.2 and 3.3.

Corollary 3.1. *If L is large enough and $Q(0) \geq L-1-\lambda\tau$, then*

$$Q(\tau) \leq Q(0) - \Delta + \frac{Q^{>L}(0)}{L} B(L,\tau) < Q(0) - \Delta + Q^{>L}(0).$$

Proof. Taking $K=0$ in Lemma 3.3, we obtain

$$\begin{aligned}
Q(\tau) &\leq Q(0) + \lambda\tau - \frac{Q^{\leq L}(0)}{L} B(L,\tau) \\
&\leq Q(0) + \lambda\tau - \frac{L-1-\lambda\tau - Q(0) + Q^{\leq L}(0)}{L} B(L,\tau) \\
&= Q(0) - \Delta_L + \frac{Q^{>L}(0)}{L} B(L,\tau) \leq Q(0) - \Delta + \frac{Q^{>L}(0)}{L} B(L,\tau)
\end{aligned}$$

with

$$\Delta_L = \left(1 - \frac{\lambda\tau+1}{L}\right) B(L,\tau) - \lambda\tau \tag{3.10}$$

increasing in L, $\Delta = \Delta_{L^*}$ and $L^* = s(\lambda,\tau)$ the smallest value that is large enough, note that $\Delta > 0$ because of (3.8). □

Corollary 3.2. *If L is large enough and $Q(0) \leq L-1-\lambda\tau$, then*

$$Q(\tau) \leq L-1-\lambda\tau - \Delta + \frac{Q^{>L}(0)}{L} B(L,\tau) < L-1-\lambda\tau - \Delta + Q^{>L}(0).$$

Proof. Taking $K=0$ in Lemma 3.3 and noting that $1 - \frac{1}{L}B(L,\tau) \geq 0$ since $B(L,\tau) \leq L$, we obtain

$$\begin{aligned}
Q(\tau) &\leq Q(0) + \lambda\tau - \frac{Q^{\leq L}(0)}{L} B(L,\tau) \\
&= Q(0)\left(1 - \frac{1}{L}B(L,\tau)\right) + \frac{Q(0) - Q^{\leq L}(0)}{L} B(L,\tau) + \lambda\tau \\
&\leq (L-1-\lambda\tau)\left(1 - \frac{1}{L}B(L,\tau)\right) + \frac{Q(0) - Q^{\leq L}(0)}{L} B(L,\tau) + \lambda\tau \\
&= L-1-\lambda\tau - \Delta_L + \frac{Q^{>L}(0)}{L} B(L,\tau) \leq L-1-\lambda\tau - \Delta + \frac{Q^{>L}(0)}{L} B(L,\tau),
\end{aligned}$$

because of (3.8). □

We now present Lemmas 3.5 and 3.6, which use Corollaries 3.1 and 3.2 and Lemmas 3.1, 3.3 and 3.4 to show that under certain conditions, the total queue mass as well as the mass above level L strictly decrease.

Lemma 3.5. *If L is large enough, $L-1-\lambda\tau \le Q(0) \le L-\lambda\tau$ and $Q^{>L}(0) < \Delta$, then*

- *$Q(\tau) < Q(0) - D$, where $D > 0$, so Q is strictly smaller at the next update,*
- *$Q^{>L}(\tau) \le Q^{>L}(0)$, so $Q^{>L}$ remains strictly smaller than Δ.*

Proof. The first statement follows from Corollary 3.1, with $D = \Delta/2 - Q^{>L}(0)/2$. The second assertion follows from Lemmas 3.1 and 3.4. □

Lemma 3.6. *If L is large enough, $Q(0) \le L-1-\lambda\tau$ and $Q^{>L}(0) < \Delta$ then*

- *$Q(\tau) < L-1-\lambda\tau$, so Q remains strictly smaller than $L-1-\lambda\tau$ at the next update,*
- *$Q^{>L}(\tau) \le Q^{>L}(0)$, so $Q^{>L}$ remains strictly smaller than Δ,*
- *$Q^{>L-1}(\tau) \le cQ^{>L-1}(0)$ if $Q^{>L-1}(0) \ge \Delta$, where $c < 1$, so $Q^{>L-1}$ is strictly decreasing by a constant factor over each update interval.*

Proof. The first statement follows from Corollary 3.2. The second assertion follows from Lemmas 3.1 and 3.4. The third statement holds for

$$c = \mathrm{e}^{-\tau} + (1-\mathrm{e}^{-\tau})\frac{Q^{>L}(0)}{\Delta} < 1$$

since the final portion of Lemma 3.3 in conjunction with Lemma 3.4 gives

$$\frac{Q^{>L-1}(\tau)}{Q^{>L-1}(0)} \le \mathrm{e}^{-\tau} + (1-\mathrm{e}^{-\tau})\frac{Q^{>L}(0)}{Q^{>L-1}(0)} \le \mathrm{e}^{-\tau} + (1-\mathrm{e}^{-\tau})\frac{Q^{>L}(0)}{\Delta}.$$

□

Lemma 3.7. *If L is large enough, $Q(0) \le L-\lambda\tau$ and $Q^{>L}(0) < \Delta$, then there exists a finite time τ_L^* such that $Q(\tau_L^*) \le L-1-\lambda\tau$ and $Q^{>L-1}(\tau_L^*) < \Delta$ (as defined in (3.9)).*

Proof. The proof is constructed by applying Lemmas 3.5 and 3.6 in succession. Since $Q(0) \le L-\lambda\tau$ and $Q^{>L}(0) < \Delta$, Lemma 3.5 can be applied, so that Q is strictly decreasing and eventually becomes smaller than $L-1-\lambda\tau$ while $Q^{>L}$ remains smaller than Δ. Note that D does not decrease after any iteration since $Q^{>L}(0)$ can only decrease. At that moment, Lemma 3.6 can be applied which shows that $Q^{>L-1}$ decreases by a constant factor over each update interval as long as $Q^{>L-1}$ is larger than Δ. The constant factor c does not increase after any iteration, and in fact only becomes smaller as $Q^{>L}(0)$ decreases. In other words, $Q^{>L-1}$ becomes smaller than Δ after finitely many updates, while Q remains smaller than $L-1-\lambda\tau$ and $Q^{>L}$ smaller than Δ. □

Proof of Proposition 3.1. The procedure in the proof of Lemma 3.7 can be performed as long as L is large enough. The left-hand side of (3.8) is increasing in L (as both factors are increasing in L), which shows that the condition (3.8) becomes tighter for smaller values of L. In fact, the mass above level L^* will vanish, where L^* is the lowest value of L which is sufficiently large for (3.8) to hold, yielding the first statement of Proposition 3.1. The latter part follows directly from Lemma 3.5.

Finally, we stress that the lemma can also be applied when the maximum initial queue length is infinite, but the mass $Q(0)=\sum_{i=0}^{\infty} z_i(0)<\infty$ is finite. In that case, one can find a value for $\bar{L}$ such that $Q^{>\bar{L}}(0)=\sum_{i=\bar{L}}^{\infty} z_i(0)<\Delta$ (as the tail of a convergent series tends to zero) and $Q(0)\le \bar{L}-\lambda\tau$. In that case, the lemma can be successively applied, starting from $\bar{L}$, which proves Proposition 3.1.

We now proceed to state the second main result in this section. Denote by y^* the fluid state with $y^*_{0,0}=1-\lambda$, $y^*_{0,1}=0$, $y^*_{1,1}=\lambda$, and $y^*_{i,j}=0$ for all $j\ge i\ge 2$.

Proposition 3.2 (No-queueing threshold for δ in SUJSQ$^{\text{det}}(\delta)$). *Suppose $\delta>\lambda/(1-\lambda)$, or equivalently, $\tau=1/\delta<(1-\lambda)/\lambda=1/\lambda-1$. Then y^* is a fixed point of the fluid-limit process at update moments in the following sense:*

(a) If $y(0)=y^*$, then $y(k\tau)=y^*$ for all $k\ge 0$;

(b) For any initial state $y(0)$ with $Q(0)<\infty$, $y(k\tau)\to y^*$ as $k\to\infty$.

Moreover, in case (a), $y_{0,0}(k\tau+t)=1-\lambda-\lambda t$, $y_{0,1}(k\tau+t)=\lambda t$, $y_{1,1}(k\tau+t)=\lambda$ for all $k\ge 0$, $t\in[0,\tau)$, and $y_{i,j}(t)=0$ for all $j\ge i\ge 2$, $t\ge 0$.
In case (b), $y_{0,0}(k\tau+t)\to 1-\lambda-\lambda t$, $y_{0,1}(k\tau+t)\to\lambda t$, $y_{1,1}(k\tau+t)\to\lambda$ for all $k\ge 0$, $t\in[0,\tau)$, and $y_{i,j}(t)\to 0$ for all $j\ge i\ge 2$, $t\ge 0$.

Loosely speaking, Proposition 3.2 implies that for $\delta\ge\lambda/(1-\lambda)$, in the long term the fraction of jobs that incur a non-zero waiting time vanishes. We note that in this regime, jobs are only sent to idle servers, which means that servers only need to send feedback whenever they are idle at an update moment. Since the fraction of idle servers in the fixed point is $1-\lambda$, a sparsified version of

SUJSQ$^{\text{det}}(\delta)$ will have a communication overhead of λ per time unit or 1 message per job.

The next lemma, whose proof is presented in Section 3.A.2, provides lower and upper bounds for the number of service completions on fluid level and the total queue mass, which play an instrumental role in the proof of Proposition 3.2. The bounds and proof arguments are similar in spirit to those of Lemma 3.2 with $K = 0$, but involve crucial refinements by additionally accounting for service completions of arriving jobs.

Lemma 3.8. *If $v_0(0) \leq 1 - \lambda$, then (i) $v_0(\tau) \leq 1 - \lambda$.*

If $v_0(0) \geq \lambda\tau$, then $\int_0^\tau [1 - v_0(s)]\mathrm{d}s$ is bounded from below by

$$\begin{aligned}&\lambda[\tau - 1 + \mathrm{e}^{-\tau}] + z_1(0)[1 - \mathrm{e}^{-\tau}] + z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}]\\ &= \lambda\tau + [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] + z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}],\end{aligned}$$

so that in view of (3.6) *(ii)*

$$Q(\tau) \leq Q(0) - [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] - z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}],$$

and

$$Q(\tau) \geq Q(0) - [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] - z_2(0)\tau,$$

so that in view of (3.6) *(iii),*

$$\begin{aligned}\int_0^\tau [1 - v_0(s)]\mathrm{d}s &\leq \lambda[\tau - 1 + \mathrm{e}^{-\tau}] + z_1(0)[1 - \mathrm{e}^{-\tau}] + z_2(0)\tau\\ &= \lambda\tau + [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] + z_2(0)\tau.\end{aligned}$$

If $v_0(0) < \lambda\tau$, then $\int_0^\tau [1 - v_0(s)]\mathrm{d}s$ is bounded from below by

$$\begin{aligned}&\lambda[\tau - 1 + \mathrm{e}^{-\tau}] + \hat{z}_1(0)[1 - \mathrm{e}^{-\tau}] + \hat{z}_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}]\\ &= \lambda\tau + [\hat{z}_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] + \hat{z}_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}],\end{aligned}$$

so that in view of (3.6) *(iv)*

$$Q(\tau) \leq Q(0) - [\hat{z}_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] - \hat{z}_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}],$$

with $\hat{z}_i(0) = \min\{z_i(0), 1 - \lambda\tau\}$, $i = 1, 2$.

We deduce that (v)

$$\begin{aligned}Q(\tau) \leq Q(0) &- \min\{1 - v_0(0) - \lambda, 1 - \lambda\tau - \lambda\}[1 - \mathrm{e}^{-\tau}]\\ &- \min\{z_2(0), 1 - \lambda\tau\}[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}],\end{aligned}$$

with $1 - \lambda\tau - \lambda > 0$.

Proof of Proposition 3.2. We first consider case (a) with $y(0) = y^*$. The fluid-limit equations can then be explicitly solved to obtain $y_{0,0}(t) = 1 - \lambda - \lambda t$, $y_{0,1}(t) = \lambda t$, $y_{1,1}(t) = \lambda$, and $y_{i,j}(t) = 0$ for all $j \geq i \geq 2$ for $t \in [0, \tau)$.

Since at an update moment $y_{0,0}(\tau) = y_{0,0}(\tau^-) + y_{0,1}(\tau^-) = 1 - \lambda$, $y_{1,1}(\tau) = y_{1,1}(\tau^-) = \lambda$, and $y_{i,j}(\tau) = y_{i,j}(\tau^-) = 0$ for all $j \geq i \geq 2$, we obtain a strictly cyclic evolution pattern with $y(k\tau) = y^*$ for all $k \geq 0$, as well as $y_{0,0}(k\tau + t) = 1 - \lambda - \lambda t$, $y_{0,1}(k\tau + t) = \lambda t$, $y_{1,1}(k\tau + t) = \lambda$ for all $k \geq 0$ $t \in [0, \tau)$, and $y_{i,j}(t) = 0$ for all $j \geq i \geq 2$, $t \geq 0$.

We now turn to case (b). First suppose that there exists a $k_0 < \infty$ such that $v_0(k_0\tau) \leq 1 - \lambda$. It then follows from statement (i) in Lemma 3.8 that $v_0(k\tau) \leq 1 - \lambda$ for all $k \geq k_0$. Moreover, in view of statement (v) in Lemma 3.8 we have $Q((k + 1)\tau) \leq Q(k\tau) - c\min\{\epsilon, \Delta\} - d\min\{z_2(k\tau), 1 - \lambda\tau\}$, with $c > 0$ and $d > 0$ when $v_0(k\tau) \leq 1 - \lambda - \epsilon$ for any $\epsilon > 0$. Thus, for any $\epsilon > 0$ it can only occur finitely many times that $v_0(k\tau) \leq 1 - \lambda - \epsilon$, and additionally for any $\delta > 0$, it can only occur finitely many times that $z_2(k\tau) \geq \delta$, because otherwise $Q(k\tau)$ would eventually fall to zero, which would contradict $v_0(k\tau) \leq 1 - \lambda$. Thus we conclude that $v_0(k\tau) \to 1 - \lambda$ and $z_2(k\tau) \to 0$, which implies that $y(k\tau) \to y^*$ as $k \to \infty$ as stated.

Now suppose that there exists no $k_0 < \infty$ such that $v_0(k_0\tau) \leq 1 - \lambda$, i.e, $v_0(k\tau) > 1 - \lambda$ for all $k \geq 1$. Solving (3.1) then gives $\frac{\mathrm{d}}{\mathrm{d}t} w_0(t) = -\frac{\mathrm{d}}{\mathrm{d}t} w_1(t) = -\lambda$ and $m(t) = 0$ for $t \in [k\tau, k\tau + v_0(k\tau)/\lambda]$, where $v_0(k\tau)/\lambda > \tau$. Lemma 3.3 with $K = 1$, $L = 1$ yields

$$Q^{>1}((k+1)\tau) \leq Q^{>1}(k\tau) - z_2(k\tau)[1 - \mathrm{e}^{-\tau}].$$

Thus we must have $z_2(k\tau) \to 0$ as $k \to \infty$, because otherwise $Q^{>1}(k\tau)$ would eventually drop below zero, which would contradict the fact that it must always be positive. Hence, for any $\epsilon > 0$, there exists k_ϵ such that $z_2(k\tau) \leq \epsilon(1 - \mathrm{e}^{-\tau})/(2\tau)$ for all $k \geq k_\epsilon$. Statement (iii) in Lemma 3.8 may then be invoked to obtain that for any $\epsilon > 0$ and $k \geq k_\epsilon$ if $v_0(k\tau) \geq 1 - \lambda + \epsilon \geq \lambda\tau$, then

$$Q((k+1)\tau) \geq Q(k\tau) + \frac{\epsilon}{2}\frac{1 - \mathrm{e}^{-\tau}}{\tau}.$$

It follows that for any $\epsilon > 0$, it can only occur finitely many times that $v_0(k\tau) \geq 1 - \lambda + \epsilon$, as $Q(k\tau)$ is bounded since $Q^{>1}(k\tau)$ is decreasing (Lemma 3.1). Thus we conclude $v_0(k\tau) \to 1 - \lambda$ and $z_2(k\tau) \to 0$ as $k \to \infty$, implying that $y(k\tau) \to y^*$ as $k \to \infty$ as stated.

□

3.4 Asynchronous updates

In this section we turn to the fluid limit for asynchronous updates. We will focus on the fluid limit for exponential update intervals, which is more tractable because all transitions are memoryless, including those involving updates. In Section 3.4.1 we provide a characterization of the fluid-limit trajectory, along with a heuristic explanation, numerical illustration and comparison with simulation. In Section 3.4.2 the fixed point of the fluid limit is determined (Proposition 3.3), which immediately shows that in stationarity queueing vanishes at fluid level for sufficiently high δ (Corollary 3.3) and also provides an upper bound for the queue length at fluid level for any given $\delta > 0$ (Corollary 3.4).

3.4.1 Fluid-limit dynamics

In the case of synchronized updates, the minimum queue estimate $m(y^N(t))$ could never decrease between successive update moments. As a result, the amount of time $\int_{t_0}^t \mathbb{1}\{m(Y^N(s)) = j\}\mathrm{d}s$ that the minimum queue length equals j in between successive update moments converges to $\int_{t_0}^t \alpha_j(s)\mathrm{d}s$, as $N \to \infty$, with $\alpha_j(t) = \mathbb{1}\{m(y(t)) = j\}$ and can be directly expressed in terms of the minimum queue estimate on fluid scale.

In contrast, with asynchronous updates, the minimum queue estimate may drop at any time when an individual server with a queue length $i < m(y^N(t))$ sends an update at time t, and becomes the only server with a queue estimate below $m(y^N(t))$. Consequently, the amount of time $\int_{t_0}^t \mathbb{1}\{m(Y^N(s)) = j\}\mathrm{d}s$ that the minimum queue length equals j no longer tends to $\int_{t_0}^t \alpha_j(s)\mathrm{d}s$ as $N \to \infty$, and may even have a positive derivative for $j < m(y(t))$, i.e., for queue values strictly smaller than the minimum queue estimate on fluid scale.

The fact that even in the limit the system may spend a non-negligible amount of time in states that are not directly visible on fluid scale severely complicates the characterization of the fluid limit. In order to handle this complication and describe the evolution of the fluid limit, it is convenient to define $u_k(t) = \delta \sum_{i=0}^{k-1}(k-i)v_i(t)$ as the fluid-scaled rate at which the dispatcher can assign jobs to servers with queue estimates below k as a result of updates, with $v_i(t) = \sum_{l=i}^{\infty} y_{i,l}(t)$ representing the fraction of servers with queue length i in fluid state y at time t as before.

We distinguish two cases, depending on whether $u_{m(t)}(t) \leq \lambda$ or not, and additionally introduce $n(t)$, defined as $n(t) = m(t)$ in case $u_{m(t)}(t) \leq \lambda$, or $n(t) = \min\{n : u_n(t) > \lambda\} \leq m(t) - 1$ otherwise. Then servers with a true queue length

$i \leq n(t)-1$ will be assigned $n(t)-i$ jobs almost immediately after an update at time t, and then have both queue length and queue estimate $n(t)$. Incoming jobs will be assigned to servers with a queue estimate at most $n(t)-1$ at rate $u_{n(t)}(t)$ and to servers with queue estimate exactly equal to $n(t)$ at rate $\zeta(t) = \lambda - u_{n(t)}(t)$.

Then the fluid limit $y(t)$ satisfies the system of differential equations

$$\begin{aligned}\frac{\mathrm{d}y_{i,j}(t)}{\mathrm{d}t} &= y_{i+1,j}(t)\mathbb{1}\{i<j\} - y_{i,j}(t)\mathbb{1}\{i>0\} + \zeta(t)q_{i-1,j-1}(t)\mathbb{1}\{i>0\} - \zeta(t)q_{i,j}(t)\\ &\quad + \delta\sum_{k=0}^{n(t)-1} v_k(t)\mathbb{1}\{i=j=n(t)\} + \delta v_i(t)\mathbb{1}\{i=j\geq n(t)\} - \delta y_{ij}(t),\end{aligned} \tag{3.11}$$

for all $i = 0,1,\ldots, j \geq n(t)$, where

$$q_{i,j}(t) = \frac{y_{i,j}(t)}{w_j(t)}\mathbb{1}\{n(t)=j\}$$

denotes the fraction of jobs assigned to a server with queue length i and queue estimate j in fluid state y among the ones that are assigned to a server with queue estimate at least $n(t)$, defined as function of the fluid state y at time t as above.

It can be checked that when $n(t) < m(t)$, the derivative of $\sum_{j=0}^{m(t)} w_j(t)$ is strictly positive, i.e., the fraction of servers with a queue estimate below $m(t)$ becomes positive, and the value of $m(t)$ instantly becomes equal to $n(t)$.

Informal outline of the derivation

We now provide an informal outline of the derivation of the fluid limit as stated in (3.11). Let $A_{i,j}(t)$, $B_{i,j}(t)$ and $S_{i,j}(t)$ denote unit-rate Poisson processes, $j \geq i \geq 0$, all independent. The system dynamics may then be represented (see [HK94; PTW07]) as

$$\begin{aligned}Y_{i,j}^N(t) = {} & Y_{i,j}^N(0) + S_{i+1,j}\left(\int_0^t Y_{i+1,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i<j\} - S_{i,j}\left(\int_0^t Y_{i,j}^N(s)\mathrm{d}s\right)\mathbb{1}\{i>0\}\\ & + A_{i-1,j-1}\left(\lambda N\int_0^t p_{i-1,j-1}(Y^N(s))\mathrm{d}s\right)\mathbb{1}\{i>0\} - A_{i,j}\left(\lambda N\int_0^t p_{i,j}(Y^N(s))\mathrm{d}s\right)\\ & + \sum_{k=j}^{\infty} B_{i,k}\left(\delta\int_0^t Y_{i,k}^N(s)\mathrm{d}s\right)\mathbb{1}\{i=j\} - B_{i,j}\left(\delta\int_0^t Y_{i,j}^N(s)\mathrm{d}s\right),\end{aligned}$$

with $p_{i,j}(Y)$ as before. Dividing by N and rewriting in terms of the fluid-scaled variables $y_{i,j}^N(t) = \frac{1}{N} Y_{i,j}^N(t)$, we obtain

$$\begin{aligned} y_{i,j}^N(t) = {} & y_{i,j}^N(0) + \frac{1}{N} S_{i+1,j}\left(N \int_0^t y_{i+1,j}^N(s)\mathrm{d}s\right) \mathbb{1}\{i < j\} \\ & - \frac{1}{N} S_{i,j}\left(N \int_0^t y_{i,j}^N(s)\mathrm{d}s\right) \mathbb{1}\{i > 0\} \\ & + \frac{1}{N} A_{i-1,j-1}\left(\lambda N \int_0^t p_{i-1,j-1}^N(Y^N(s))\right) \mathbb{1}\{i > 0\} - \frac{1}{N} A_{i,j}\left(\lambda N \int_0^t p_{i,j}^N(Y^N(s))\mathrm{d}s\right) \\ & + \frac{1}{N} \sum_{k=j}^{\infty} B_{i,k}\left(\delta N \int_0^t y_{i,k}^N(s)\mathrm{d}s\right) \mathbb{1}\{i = j\} - \frac{1}{N} B_{i,j}\left(\delta N \int_0^t y_{i,j}^N(s)\mathrm{d}s\right). \end{aligned} \tag{3.12}$$

Now introduce

$$\tilde{S}_{k,l}(u) := S_{k,l}(u) - u, \quad \tilde{A}_{k,l}(u) := A_{k,l}(u) - u, \quad \tilde{B}_{k,l}(u) := B_{k,l}(u) - u,$$

and observe that $\tilde{S}_{k,l}(\cdot)$, $\tilde{A}_{k,l}(\cdot)$ and $\tilde{B}_{k,l}(\cdot)$ are martingales. By standard arguments it can be shown that

$$\frac{1}{N} \tilde{S}_{k,l}\left(N \int_0^t y_{k,l}^N(s)\mathrm{d}s\right), \quad \frac{1}{N} \tilde{A}_{k,l}\left(\lambda N \int_0^t p_{k,l}(Y^N(s))\mathrm{d}s\right), \quad \frac{1}{N} \tilde{B}_{k,l}\left(\delta N \int_0^t y_{k,l}^N(s)\mathrm{d}s\right)$$

each converge to zero as $N \to \infty$.

Adopting time-scale separation arguments [HK94; PTW07], it can be established that

$$\lambda \int_0^t p_{i,j}(Y^N(s))\mathrm{d}s \to \int_0^t \alpha_{i,j}(s)\mathrm{d}s$$

as $N \to \infty$, where the coefficients $\alpha_{i,j}(\cdot)$ satisfy

$$\alpha_{i,j}(t) = \begin{cases} 0 & i < j < n(t), \\ \lambda \pi_j(t) & i = j < n(t), \\ \lambda \frac{y_{i,j}(t)}{w_j(y(t))} \pi_{n(t)}(t) & i \le j = n(t), \\ 0 & i \le j > n(t). \end{cases}$$

The coefficients $\pi_j(t)$ may be interpreted as the fraction of time that the pre-limit minimum queue estimate equals $j \le n(t)$ when the minimum queue estimate at fluid level is $n(t)$, and satisfy the relationship

$$\lambda \pi_j(t) = \lambda \pi_{j-1}(t) + \delta v_j(y(t))$$

for all $j = 1, \dots, n(t) - 1$, along with the normalization condition $\sum_{j=0}^{n(t)} \pi_j(t) = 1$.

Thus, we obtain

$$\lambda \pi_j(t) = \delta \sum_{k=0}^{j} v_k(y(t)),$$

for all $j = 0, \dots, n(t) - 1$, and

$$\begin{aligned}\lambda \pi_{n(t)}(t) &= \lambda(1 - \sum_{j=0}^{n(t)-1} \pi_j(t)) = \lambda - \delta \sum_{j=0}^{n(t)-1} \sum_{k=0}^{j} v_k(y(t)) \\ &= \lambda - \delta \sum_{i=0}^{n(t)-1} (n(t) - i) v_i(y(t)) = \zeta(y(t)).\end{aligned}$$

We deduce that

$$\alpha_{i,j}(t) = \mathbb{1}\{i = j\} \delta \sum_{k=0}^{j} v_k(y(t)) \mathbb{1}\{j < n(t)\} + q_{i,j}(y(t)\zeta(y(t)),$$

with $q_{i,j}(y) = \frac{y_{i,j}}{w_j(y)} \mathbb{1}\{n(y) = j\}$ as before, yielding

$$\begin{aligned}&\int_0^t \alpha_{i-1,j-1}(s) \mathrm{d}s - \int_0^t \alpha_{i,j}(s) \mathrm{d}s \\ &= \int_0^t q_{i-1,j-1}(y(s))\zeta(y(s)) \mathrm{d}s - \int_0^t q_{i,j}(y(s))\zeta(y(s)) \mathrm{d}s \\ &+ \mathbb{1}\{i = j\} \delta \sum_{k=0}^{j-1} v_k(y(s)) \mathbb{1}\{j = n(s)\} \mathrm{d}s - \mathbb{1}\{i = j\} \delta \int_0^t v_j(y(s)) \mathbb{1}\{j < n(s)\} \mathrm{d}s.\end{aligned}$$

We obtain

$$\begin{aligned}&\lambda \int_0^t p_{i-1,j-1}(Y^N(s)) \mathrm{d}s \mathbb{1}\{i > 0\} - \lambda \int_0^t p_{i,j}(Y^N(s)) \mathrm{d}s \to \\ &\int_0^t q_{i-1,j-1}(y(s))\zeta(y(s)) \mathrm{d}s \mathbb{1}\{i > 0\} - \int_0^t q_{i,j}(y(s))\zeta(y(s)) \mathrm{d}s \\ &+ \mathbb{1}\{i = j\} \delta \sum_{k=0}^{j-1} v_k(y(s)) \mathbb{1}\{j = n(y(s))\} \mathrm{d}s - \mathbb{1}\{i = j\} \delta \int_0^t v_j(y(s)) \mathbb{1}\{j < n(y(s))\} \mathrm{d}s,\end{aligned} \tag{3.13}$$

as $N \to \infty$, with $\zeta(y) = \lambda - \delta \sum_{l=0}^{n(y)} (n(y) - l) v_l(y)$ and $q_{i,j}(y) = \frac{y_{i,j}}{w_j(y)} \mathbb{1}\{n(y) = j\}$ as defined earlier.

Taking the limit for $N \to \infty$ in (3.12), and noting that

$$\begin{aligned}
&\mathbb{1}\{i = j\}\delta \int_0^t v_i(y(s))\mathrm{d}s - \mathbb{1}\{i = j\}\delta \int_0^t v_j(y(s))\mathbb{1}\{j < n(y(s))\}\mathrm{d}s \\
&= \delta \int_0^t v_i(y(s))\mathbb{1}\{i = j \geq n(y(s))\}\mathrm{d}s,
\end{aligned}$$

we conclude that any (weak) limit $\{y_{i,j}(t)\}_{t\geq 0}$ of the sequence $(\{y_{i,j}^N(t)\}_{t\geq 0})_{N\geq 1}$ must satisfy

$$\begin{aligned}
y_{i,j}(t) &= y_{i,j}(0) + \int_0^t y_{i+1,j}(s)\mathrm{d}s \mathbb{1}\{i < j\} - \int_0^t y_{i,j}(s)\mathrm{d}s \mathbb{1}\{i > 0\} \\
&+ \lambda \int_0^t q_{i-1,j-1}(y(s))\zeta_{j-1}(y(s))\mathrm{d}s \mathbb{1}\{i > 0\} \\
&- \lambda \int_0^t q_{i,j}(y(s))\zeta_j(y(s))\mathrm{d}s + \delta \sum_{k=0}^{i-1} \int_0^t v_k(y(s))\mathbb{1}\{i = j = n(y)\}\mathrm{d}s \\
&+ \delta \int_0^t v_i(y(s))\mathbb{1}\{i = j \geq n(y(s))\}\mathrm{d}s - \delta \int_0^t y_{i,j}(s)\mathrm{d}s,
\end{aligned}$$

with $y_{i,j}(0) = y_{i,j}^\infty$. Rewriting the latter integral equation in differential form yields (3.11).

Interpretation

The above system of differential equations may be intuitively interpreted as follows. The first two terms correspond to service completions at servers with $i+1$ and i jobs, just like in (3.1). The third and fourth terms account for job assignments to servers with a queue estimate $m(t)$. The third term captures the resulting increase in the fraction of servers with queue estimate $m(t)+1$, while the fourth term captures the corresponding decrease in the fraction of servers with queue estimate $m(t)$.

The final three terms in (3.11) correspond to the updates from servers received at a rate δ. The fifth term represents the increase in the fraction of servers with queue estimate $n(t)$ due to updates from servers with a queue length $k \leq n(t) - 1$ which are almost immediately being assigned $n(t) - k$ jobs and then have both queue length $i = n(t)$ and queue estimate $j = n(t)$. The sixth term represents the increase in the fraction of servers with a queue estimate $j = n(t)$ or larger due to updates from servers with a queue length $i = j$. The final term represents the decrease in the number of servers with queue length i and queue estimate j due to updates.

Even though a non-zero fraction of the jobs are assigned to servers with a queue estimate below $n(t)$, these events are not directly visible at fluid level, and only implicitly enter the fluid limit through the thinned arrival rate $\zeta(t)$.

Summing the equations (3.11) over $i = 0, 1, \dots, j$ yields

$$\frac{\mathrm{d}w_j(t)}{\mathrm{d}t} = \zeta(t)[\mathbb{1}\{n(t) = j-1\} - \mathbb{1}\{n(t) = j\}] + \delta[v_j(t) - w_j(t)]\mathbb{1}\{j \geq n(t)\},$$

reflecting that servers with queue estimate $n(t)$ are assigned jobs, and thus flipped into servers with queue estimate $n(t)+1$, at rate $\zeta(t)$, and that servers with a queue estimate $j \geq n(t)$ are created at an effective rate $\delta[v_j(t) - w_j(t)]$ as a result of updates.

Numerical illustration and comparison with simulation

Figures 3.6a-3.7b show the fluid-limit trajectories $y(t)$ as governed by the differential equations in (3.11) for AUJSQ$^{\text{exp}}(\delta)$, through stochastic simulation for a system with $N = 1000$ servers and averaged over 10 runs. Once again, the simulation results are nearly indistinguishable from the fluid-limit trajectories.

In contrast to the synchronized variants in Figures 3.2a, 3.2b, 3.3a and 3.3b, the trajectories do not oscillate, but approach stable values, corresponding to the fixed point of the fluid-limit equations (3.11) which we will analytically determine in Proposition 3.3. In Figures 3.6a and 3.6b where $\delta = 0.85$ is relatively low, we observe once again that $w_2(y(t)) = w_2(t)$ and $v_2(y(t)) = v_2(t)$ become strictly positive. In Figures 3.7a and 3.7b where $\delta = 2.5$ is sufficiently large, all servers have either zero or one jobs in the limit, indicating that no queueing occurs.

Qualitatively similar results are observed for AUJSQ$^{\text{det}}(\delta)$, where the updates occur at strictly regular moments. The results are displayed in Figures 3.8a, 3.8b, 3.9a and 3.9b, for a system with $N = 400$ servers and $\lambda = 0.7$.

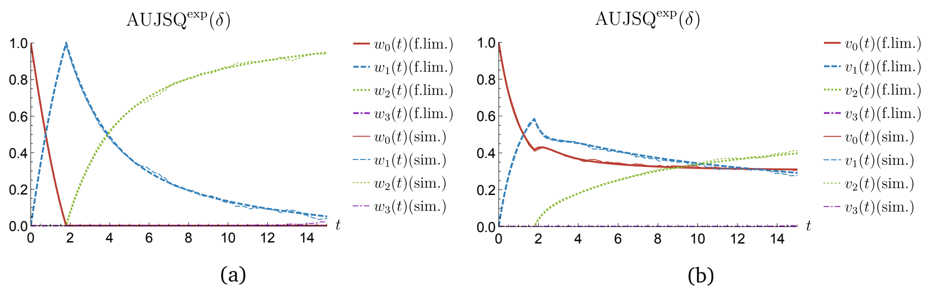


Figure 3.6: Numerical emulation of the fluid limit for AUJSQ$^{\text{exp}}(0.85)$ and $\lambda = 0.7$, accompanied by simulation results with $N = 1000$, averaged over 10 runs.

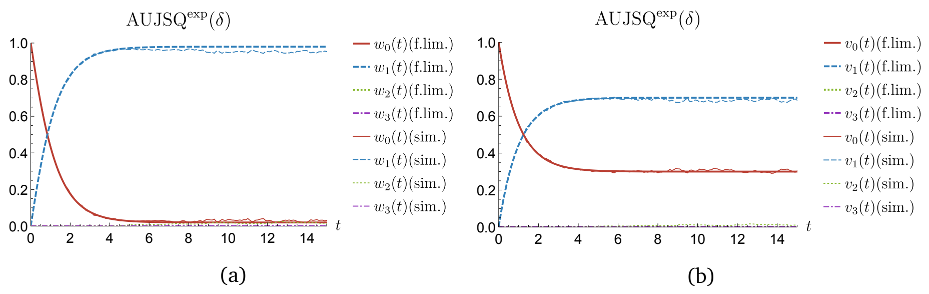


Figure 3.7: Numerical emulation of the fluid limit for AUJSQ$^{\text{exp}}(2.5)$ and $\lambda = 0.7$, accompanied by simulation results with $N = 1000$, averaged over 10 runs.

3.4.2 Fixed-point analysis

The next proposition identifies the fixed point of the fluid-limit equations (3.11) in terms of m^*, defined as

$$\begin{aligned} m^* = m(\lambda,\delta) &= \min\left\{m : \lambda < 1 - \left(\frac{1}{1+\delta}\right)^{m+1}\right\} \\ &= \max\left\{m : \lambda \geq 1 - \left(\frac{1}{1+\delta}\right)^{m}\right\} = \left\lfloor -\frac{\log(1-\lambda)}{\log(1+\delta)}\right\rfloor, \end{aligned} \tag{3.14}$$

which may be interpreted as the minimum queue estimate at fluid level in stationarity. For compactness, define $a = \frac{1}{1+\delta}$ and $b = \frac{1}{1+\delta+\nu}$.

Proposition 3.3 (Fixed point for AUJSQ$^{\text{exp}}(\delta)$)**.** *The fixed point of the fluid limit* (3.11) *is given by*

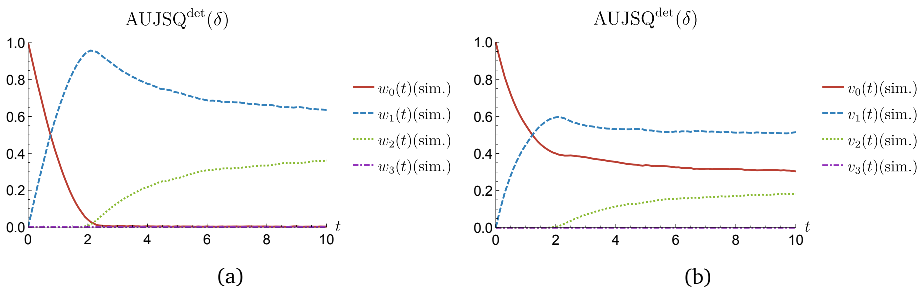


Figure 3.8: Simulation results for $\mathrm{AUJSQ}^{\mathrm{det}}(0.85)$.

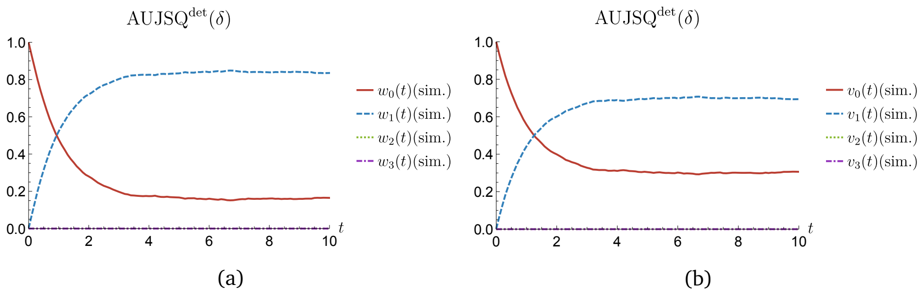


Figure 3.9: Simulation results for $\mathrm{AUJSQ}^{\mathrm{det}}(2.5)$.

$$
\begin{aligned}
y^*_{0,m^*} &= \frac{ab^{m^*-1}\delta}{(1+\nu)(\delta+\nu)},\\
y^*_{i,m^*} &= \frac{ab^{m^*-i}\delta}{1+\nu}, \qquad i = 1,\dots,m^*,\\
y^*_{0,m^*+1} &= a^{m^*+1} - \frac{a^2b^{m^*-1}\delta}{(1+\nu)(\delta+\nu)},\\
y^*_{1,m^*+1} &= \delta\left(a^{m^*+1} - \frac{a^2b^{m^*-1}\delta}{(1+\nu)(\delta+\nu)}\right),\\
y^*_{i,m^*+1} &= \delta\left(a^{m^*+2-i} - \frac{ab^{m^*+1-i}}{1+\nu}\right), \quad i = 2,\dots,m^*+1,
\end{aligned}
$$

and $y^*_{i,j} = 0$ *when* $j \neq m^*, m^* + 1$*, where* $\nu \geq 0$ *is the unique solution of the equation* $y_{0,m^*} + y_{0,m^*+1} = 1 - \lambda$*, i.e.,*

$$h(\nu) = a^{m^*+1} + \frac{a^2 b^{m^*-1} \delta^2}{(1+\nu)(\delta+\nu)} = 1 - \lambda. \tag{3.15}$$

In particular, if $\lambda = 1 - (1+\delta)^{-m^*}$*, i.e.,*

$$m^* = -\frac{\log(1-\lambda)}{\log(1+\delta)}, \tag{3.16}$$

then $\nu = 0$*, so*

$$\begin{aligned} y^*_{0,m^*} &= \left(\frac{1}{1+\delta}\right)^{m^*} = 1 - \lambda, \\ y^*_{i,m^*} &= \delta \left(\frac{1}{1+\delta}\right)^{m^*+1-i} = \left(1 - (1-\lambda)^{1/m^*}\right)(1-\lambda)^{\frac{m^*-i}{m^*}}, \quad i = 1, \dots, m^*, \end{aligned}$$

and $y^*_{i,m^*+1} = 0$ *for all* $i = 0, \dots, m^* + 1$.

Note that $h(u)$ is strictly decreasing in ν, with $\lim_{\nu \downarrow 0} h(\nu) = (\frac{1}{1+\delta})^{m^*} \leq 1 - \lambda$, and $\lim_{\nu \to \infty} h(\nu) = (\frac{1}{1+\delta})^{m^*+1} < 1 - \lambda$, ensuring that $\nu \geq 0$ exists and is unique.

The result of Proposition 3.3 is obtained by setting the derivatives in (3.11) equal to zero, observing that $w_j(y^*) = 0$ for all $j \neq m^*, m^* + 1$, and then solving the resulting equations. The detailed proof arguments are presented in Section 3.B.

Corollary 3.3 (No-queueing threshold for δ in AUJSQ$^{\text{exp}}(\delta)$). *If the update frequency* $\delta \geq \lambda/(1-\lambda)$*, then* $y^*_{i,j} = 0$ *for all* $j \geq 2$*, implying that queueing vanishes at fluid level in stationarity.*

Corollary 3.3 immediately follows from (3.14) and Proposition 3.3, in which $m^* = 0$ in case $\delta \geq \lambda/(1-\lambda)$ so that only $y^*_{0,0}$, $y^*_{0,1}$ and $y^*_{1,1}$ are strictly positive. In case of equality we have the scenario described in the last part of Proposition 3.3 where $y^*_{i,2} = 0$ for all i. The arguments for interchanging the many-server ($N \to \infty$) and stationary ($t \to \infty$) limits are beyond the scope of the manuscript, as mentioned in Section 1.3.2. Corollary 3.3 implies that for $\delta \geq \lambda/(1-\lambda)$, the mean stationary waiting time under AUJSQ$^{\text{exp}}(\delta)$ vanishes as $N \to \infty$.

Proposition 3.3 also yields an upper bound for the queue length at fluid level as stated in the next corollary.

Corollary 3.4 (Bounded queue length for AUJSQ$^{\text{exp}}(\delta)$). *The queue length at fluid level in stationarity has bounded support on* $\{0,\dots, m(\lambda,\delta)+1\}$ *for any* $\lambda < 1$ *and* $\delta > 0$.

First of all, note that in order for queueing to vanish, it is required that $m(\lambda,\delta) = 0$ or $m(\lambda,\delta) = 1$ and $\nu = 0$, i.e., $\delta \geq \lambda/(1-\lambda)$, which coincides with the threshold for δ in SUJSQ$^{\text{det}}(\delta)$ as identified in Proposition 3.2. Also, the upper bound $m(\lambda,\delta)+1$ tends to infinity as δ approaches zero, reflecting that for any fixed arrival rate λ, even arbitrarily low, the maximum queue length grows without bound as the update frequency vanishes.

At the same time, for any positive $\delta > 0$, $m(\lambda,\delta)$ is finite for any fixed $\lambda < 1$, and only grows as $\log(1/(1-\lambda))$ as $\lambda \uparrow 1$ rather than $1/(1-\lambda)$ as in the absence of any queue feedback. Thus, even an arbitrarily low update frequency ensures that the queue length has bounded support and behaves far more benignly in a high-load regime at fluid level. This powerful property resembles an observation in [TX12; TX13] in the context of a dynamic scheduling problem where even a minuscule degree of resource pooling yields a fundamentally different behavior on fluid scale.

Number of jobs in the system

The average queue length in the fixed point $\tilde{q}$ is

$$\begin{aligned}
\tilde{q} &= \sum_{i=1}^{m^*} i y^*_{i,m^*} + \sum_{i=1}^{m^*+1} i y^*_{i,m^*+1} = \frac{\delta + a^{m^*+1} + \delta m^* - 1}{\delta} + \frac{a^2\delta\left(-\delta + b^{m^*} - 1\right)}{b(1+\nu)(\delta+\upsilon)} \\
&\overset{(3.15)}{=} m^* + 1 + \frac{a^{m^*+1}-1}{\delta} + \frac{1-\lambda-a^{m^*+1}}{\delta} - \frac{a(1+\delta+\nu)\delta}{(1+\nu)(\delta+\nu)} \qquad (3.17) \\
&= m^* + 1 - \lambda/\delta - \frac{(1+\delta+\nu)\delta}{(1+\delta)(1+\nu)(\delta+\nu)} \geq m^* - \lambda/\delta.
\end{aligned}$$

In case of (3.16), the average queue length is simply $\tilde{q} = m^* + 1 - \lambda/\delta - 1 = m(\lambda,\delta) - \lambda/\delta$, reflecting that the average number of job arrivals equals the average number of job completions over the course of an update interval, starting with $m^* = m(\lambda,\delta)$ jobs.

Figure 3.10 plots the average queue length in the fixed point given by (3.17) as function of the update frequency δ for $\lambda = 0.8$. We observe that the average queue length monotonically decreases with the update frequency, as expected, and is indeed contained between $m^* - \lambda/\delta$ and $m^* + 1 - \lambda/\delta$.

It then follows from the definition of $m^* = m(\lambda,\delta)$ that $\tilde{q} \to \infty$ as $\delta \downarrow 0$ for any $\lambda < 1$, which indicates that AUJSQ$^{\text{det}}(\delta)$ may perform arbitrarily badly in the ultra-low feedback regime, confirming the observations in Section 3.2.1.

Bound on the queue length for AUJSQ$^{\text{det}}(\delta)$

As noted earlier, the fluid-limit trajectory for AUJSQ$^{\text{det}}(\delta)$ involves a measure-valued process and is difficult to describe. However, in a similar spirit as for AUJSQ$^{\text{exp}}(\delta)$, the value of m^* can be characterized as the largest integer for which

$$\left(\sum_{i=1}^{m} i\frac{(1/\delta)^i}{i!}\mathrm{e}^{-1/\delta} + m\sum_{i=m+1}^{\infty}\frac{(1/\delta)^i}{i!}\mathrm{e}^{-1/\delta}\right) \le \lambda/\delta,$$

expressing that the average number of job arrivals should be larger than or equal to the average number of job completions over the course of an update interval, starting with m^* jobs. While the above equation cannot easily be solved in closed form, it is not difficult to show that the inequality is weaker than (3.14), i.e., the value of m^* is lower than for AUJSQ$^{\text{exp}}(\delta)$, confirming the superiority of AUJSQ$^{\text{det}}(\delta)$ observed in the simulation results in Section 3.2.1. It is further worth observing the strong similarity of the above inequality with Proposition 3.1 governing the queue length upper bound for SUJSQ$^{\text{det}}(\delta)$.

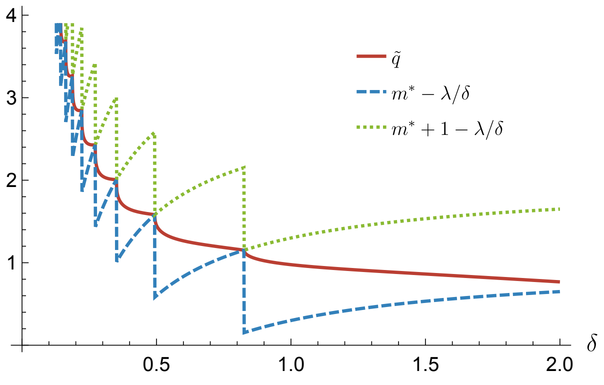


Figure 3.10: Values of $\tilde{q}$ for $\lambda = 0.7$ and different values of δ.

3.5 Conclusion

We have introduced and analyzed a novel class of hyper-scalable load balancing algorithms that only involve minimal communication overhead and yet deliver excellent performance. In the proposed schemes, the various servers provide occasional queue status notifications to guide the dispatcher in directing incoming jobs to relatively short queues. There are more hyper-scalable schemes that could be of interest. Another hyper-scalable scheme will be introduced in Chapter 4, in which the timer that governs the updates only starts when a server becomes ‘full’. Another interesting class of schemes would be one where the queue estimate is not an upper-bound but mimics the expected value of the queue length. These schemes are left as interesting topics for further research.

We have demonstrated that the schemes markedly outperform JSQ(d) policies with a comparable overhead, and can drive the waiting time to zero in the many-server limit with just one message per job. The proposed schemes show their core strength and outperform sparsified JIQ versions in the sparse feedback regime with less than one message per job, which is particularly pertinent from a scalability viewpoint.

[AD20] presents the $SQ(d,N)$ algorithm, which is similar to the JSQ(d) algorithm, and also similar to our hyper-scalable algorithms. Instead of periodic updates as we have seen in this chapter, d randomly selected queues are probed when a job arrives in $SQ(d,N)$. AUJSQ$^{\text{exp}}(\delta)$ is the closest to this algorithm, since in $SQ(d,N)$ the arrival process basically acts as a timer. The no-queueing threshold $\delta \geq \lambda/(1-\lambda)$ in Corollary 3.3 matches with the threshold $\lambda < 1 - 1/d$ that $SQ(d,N)$ has, since the communication overhead of $SQ(d,N)$ equals $\delta = \lambda d$. The same comparison holds for the support of the queue lengths.

In order to further explore the performance in the many-server limit, we investigated fluid limits for synchronized as well as asynchronous exponential update intervals. We used the fluid limits to obtain upper bounds for the stationary queue length as function of the load and update frequency. We also revealed a striking dichotomy in the ultra-low feedback regime where the mean waiting time tends to a constant in the synchronized case, but grows without bound in the asynchronous case. Extensive simulation experiments are conducted to support the analytical results, and indicate that the fluid-limit asymptotics are remarkable accurate.

We have adopted common Markovian assumptions, and it could be interesting to extend the results to non-exponential and possibly heavy-tailed distributions. In Chapter 5 we will decrease the communication overhead even further, by pursuing schemes that may dynamically suppress updates or selectively refrain from updates at pre-scheduled epochs to convey implicit information.

3.A Proofs of Section 3.3.2

3.A.1 Proof of Lemma 3.2

Let $\tilde{y}_{i,j}(t)$, $i = 0,1,\dots,j$, $j \geq 0$, be the solution to the fluid-limit equation (3.1) with $\lambda = 0$, i.e.,

$$\frac{\mathrm{d}\tilde{y}_{i,j}(t)}{\mathrm{d}t} = \tilde{y}_{i+1,j}(t)\mathbb{1}\{i < j\} - \tilde{y}_{i,j}(t)\mathbb{1}\{i > 0\},$$

with initial conditions $\tilde{y}_{i,i}(0) = v_i(0)$ and $\tilde{y}_{i,j}(0) = 0$ for all $j \geq i+1$, $i \geq 0$. The solution $\tilde{y}_{i,j}(t)$ may be interpreted as the fluid limit in the absence of any arrivals, and it is easily verified that

$$\tilde{y}_{i,j}(t) = v_j(0)\frac{t^{j-i}}{(j-i)!}\mathrm{e}^{-t},$$

for all $i = 1,2,\dots,j$, and

$$\tilde{y}_{0,j}(t) = v_j(0)\sum_{k=j}^{\infty}\frac{t^k}{k!}\mathrm{e}^{-t},$$

$j \geq 0$. Further introduce

$$\tilde{v}_i(t) = \sum_{j=i}^{\infty}\tilde{y}_{i,j}(t), \quad \tilde{z}_k(t) = \sum_{i=k}^{\infty}\tilde{v}_i(t), \quad \tilde{Q}^{>K}(t) = \sum_{k=K+1}^{\infty}\tilde{z}_k(t),$$

and note from (3.4) that

$$\frac{\mathrm{d}\tilde{Q}^{>K}(t)}{\mathrm{d}t} = -\tilde{z}_{K+1}(t). \tag{3.18}$$

We will first establish that $\tilde{z}_k(t) \leq z_k(t)$ for all $k \geq 0$, $t \in [0,\tau)$, reflecting that the fraction of servers with queue length k or larger on fluid scale is no less than what it would be in the absence of any arrivals. Suppose that were not the case, and let $t_0 \in [0,\tau)$ be the first time when that inequality is about to be violated for some $k_0 > 0$. Then we must have $z_{k_0}(t_0) = \tilde{z}_{k_0}(t_0)$, implying $v_{k_0}(t_0) =$

$z_{k_0}(t_0) - z_{k_0+1}(t_0) \le \tilde{z}_{k_0}(t_0) - \tilde{z}_{k_0+1}(t_0) = \tilde{v}_{k_0}(t_0)$, since $\tilde{z}_{k_0+1}(t_0) \le z_{k_0+1}(t_0)$. Now observe that

$$\frac{\mathrm{d}z_{k_0}(t)}{\mathrm{d}t}\,|_{t=t_0} = -v_{k_0}(t_0) + \lambda \sum_{j=k_0-1}^{\infty} p_{k_0-1,j}(t_0) \ge -v_{k_0}(t_0),$$

while

$$\frac{\mathrm{d}\tilde{z}_{k_0}(t)}{\mathrm{d}t}\,|_{t=t_0} = -\tilde{v}_{k_0}(t_0) \le -v_{k_0}(t_0) \le \frac{\mathrm{d}z_{k_0}(t)}{\mathrm{d}t}.$$

Hence $z_{k_0}(t)$ cannot fall below $\tilde{z}_{k_0}(t)$ at (just after) t_0, contradicting the initial supposition in which t_0 would be the first time that the inequality is about to be violated.

Invoking (3.18), we obtain

$$\begin{aligned}
\int_{s=0}^{t} z_{K+1}(s)\mathrm{d}s \ge \int_{s=0}^{t} \tilde{z}_{K+1}(s)\mathrm{d}s &= Q^{>K}(0) - \tilde{Q}^{>K}(t) \\
&= Q^{>K}(0) - \sum_{k=K+1}^{\infty} (k-K)\tilde{v}_k(t) \\
&= Q^{>K}(0) - \sum_{k=K+1}^{\infty} (k-K)\sum_{l=k}^{\infty} \tilde{y}_{k,l}(t) \\
&= Q^{>K}(0) - \sum_{k=K+1}^{\infty} (k-K)\sum_{l=k}^{\infty} v_l(0)\frac{t^{l-k}}{(l-k)!}\mathrm{e}^{-t} \\
&= Q^{>K}(0) - \sum_{l=K+1}^{\infty} v_l(0)\sum_{k=K+1}^{l} (k-K)\frac{t^{l-k}}{(l-k)!}\mathrm{e}^{-t} \\
&= Q^{>K}(0) - \sum_{l=K+1}^{\infty} v_l(0)\sum_{m=0}^{l-K-1} (l-K-m)\frac{t^{m}}{m!}\mathrm{e}^{-t} \\
&= Q^{>K}(0) - \sum_{l=K+1}^{\infty} v_l(0)A(l-K,t) \\
&= Q^{>K}(0) - \sum_{l=1}^{\infty} v_{K+l}(0)A(l,t).
\end{aligned}$$

Now observe that

$$\begin{aligned}
\mathrm{e}^t A(L,t) &= \sum_{l=0}^{L}(L-l)\frac{t^l}{l!} = \sum_{l=0}^{L}\frac{L-l}{L+1-l}(L+1-l)\frac{t^l}{l!}\\
&\le \sum_{l=0}^{L}\frac{L}{L+1}(L+1-l)\frac{t^l}{l!} = \frac{L}{L+1}\sum_{l=0}^{L}(L+1-l)\frac{t^l}{l!}\\
&= \frac{L}{L+1}\sum_{l=0}^{L+1}(L+1-l)\frac{t^l}{l!} = \frac{L}{L+1}\mathrm{e}^t A(L+1,t),
\end{aligned}$$

or equivalently,

$$\frac{A(L,t)}{L} \le \frac{A(L+1,t)}{L+1},$$

which may be interpreted from the fact that the expected fraction of jobs that remain after a period of length t is smaller with an initial queue of size L than $L+1$. Thus, $A(l,t) \le \frac{l}{L}A(L,t)$ for all $l \le L$. Also,

$$A(L,t) = \sum_{l=0}^{L}(L-l)\frac{t^l}{l!}\mathrm{e}^{-t} = \sum_{l=0}^{L-1}(L-1-l)\frac{t^l}{l!}\mathrm{e}^{-t} + \sum_{l=0}^{L-1}\frac{t^l}{l!}\mathrm{e}^{-t} \le A(L-1,t)+1,$$

so that $A(l,t) \le l-L+A(L,t)$ for all $l \ge L+1$. We obtain

$$\begin{aligned}
&\sum_{l=1}^{\infty} v_{K+l}(0)A(l,t) = \sum_{l=1}^{L} v_{K+l}(0)A(l,t) + \sum_{l=L+1}^{\infty} v_{K+l}(0)A(l,t)\\
&\le \sum_{l=1}^{L} v_{K+l}(0)\frac{l}{L}A(L,t) + \sum_{l=L+1}^{\infty} v_{K+l}(0)[l-L+A(L,t)]\\
&= \frac{1}{L}\left[\sum_{l=1}^{L} l v_{K+l}(0) + \sum_{l=L+1}^{\infty} v_{K+l}(0)\right]A(L,t) + \sum_{l=L+1}^{\infty}(l-L)v_{K+l}(0)\\
&= Q^{>K+L}(0) + \frac{1}{L}\left[Q^{\le K+L}(0) - Q^{\le K}(0)\right]A(L,t),
\end{aligned}$$

yielding

$$\begin{aligned}
&Q^{>K}(0) - \sum_{l=1}^{\infty} v_{K+l}(0)A(l,t) \ge Q^{>K}(0) - Q^{>K+L}(0) - \frac{1}{L}\left[Q^{\le K+L}(0) - Q^{\le K}(0)\right]A(L,t)\\
&= Q^{\le K+L}(0) - Q^{\le K}(0) - \frac{1}{L}\left[Q^{\le K+L}(0) - Q^{\le K}(0)\right]A(L,t)\\
&= \left[Q^{\le K+L}(0) - Q^{\le K}(0)\right]\left[1 - \frac{A(L,t)}{L}\right].
\end{aligned}$$

3.A.2 Proof of Lemma 3.8

Just like in the proof of Lemma 3.2, let $\tilde{y}_{i,j}(t)$, $i = 0,1,\dots,j$, $j \geq 0$, be the solution to the fluid-limit equation (3.1) with $\lambda = 0$, i.e.,

$$\frac{\mathrm{d}\tilde{y}_{i,j}(t)}{\mathrm{d}t} = \tilde{y}_{i+1,j}(t)\mathbb{1}\{i < j\} - \tilde{y}_{i,j}(t)\mathbb{1}\{i > 0\},$$

but now with initial conditions such that $\tilde{z}_i(0) \leq z_i(0)$ for all $i \geq 1$, with $\tilde{z}_i(t) = \sum_{k=i}^{\infty} \tilde{v}_k(t)$ and $\tilde{v}_k(0) = \tilde{y}_{k,k}(0)$, and $\tilde{y}_{i,j}(0) = 0$ for all $j \geq i+1$, $i \geq 0$. As before, $\tilde{y}_{i,j}(t)$ may be interpreted as the fluid limit in the absence of any arrivals, and it is easily verified that

$$\tilde{y}_{i,j}(t) = \tilde{v}_j(0)\frac{t^{j-i}}{(j-i)!}\mathrm{e}^{-t},$$

for all $i = 1,2,\dots,j$, and

$$\tilde{y}_{0,j}(t) = \tilde{v}_j(0)\sum_{k=j}^{\infty}\frac{t^k}{k!}\mathrm{e}^{-t},$$

$j \geq 0$. Further let $y^0_{0,1}(t)$, $y^0_{1,1}(t)$ be solutions to the system of differential equations

$$\begin{aligned}\frac{\mathrm{d}y^0_{0,1}(t)}{\mathrm{d}t} &= y^0_{1,1}(t)\\ \frac{\mathrm{d}y^0_{1,1}(t)}{\mathrm{d}t} &= \lambda\mathbb{1}\{t \leq t_0\} - y^0_{1,1}(t),\end{aligned}$$

with $t_0 = \min\{\tilde{v}_0(0)/\lambda, T\}$ and initial conditions $y^0_{0,1}(0) = y^0_{1,1}(0) = 0$.

It is easily verified that

$$y^0_{0,1}(t) = \begin{cases} \lambda[t - 1 + \mathrm{e}^{-t}] & t \in [0, t_0], \\ \lambda[t_0 - \mathrm{e}^{-(t-t_0)} + \mathrm{e}^{-t}] & t \in [t_0, \tau], \end{cases}$$

$$y^0_{1,1}(t) = \begin{cases} \lambda[1 - \mathrm{e}^{-t}] & t \in [0, t_0], \\ \lambda[\mathrm{e}^{-(t-t_0)} - \mathrm{e}^{-t}] & t \in [t_0, \tau]. \end{cases}$$

The variable $y^0_{0,1}(t)$ may be interpreted as the fraction of servers with queue length 0 at time 0, queue length 0 at time t and queue estimate 1 at time t, i.e., which have been assigned an arriving job and completed that job by time t. Likewise, $y^0_{1,1}(t)$ may be interpreted as the fraction of servers with queue length 0 at time 0, queue length 1 at time t and queue estimate 1 at time t, i.e., which have been assigned an arriving job that remains to completed by time t.

In a similar fashion as in the proof of Lemma 3.2, it can be established that $\tilde{z}_1(t) + y^0_{1,1}(t) \leq z_1(t)$ and $\tilde{z}_i(t) \leq z_i(t)$ for all $i \geq 2$ and $t \in [0, T]$.

To prove statement (i), consider $\tilde{z}_1(0) = \min\{z_1(0), 1 - \lambda\tau\} \geq \min\{1 - v_0(0), 1 - \lambda\tau\} \geq \min\{\lambda, 1 - \lambda\tau\} = \lambda$, and $\tilde{z}_k(0) = 0$ for all $k \geq 2$. Noting that $\tilde{v}_0(0) = 1 - \tilde{z}_1(0) \geq \lambda\tau$ yields $t_0 = \tau$, and thus $y^0_{1,1}(\tau) = \lambda(1 - \mathrm{e}^{-\tau})$. Also, $\tilde{z}_1(\tau) = \tilde{y}_{1,1}(\tau) = \tilde{v}_1(0)\mathrm{e}^{-\tau} = \tilde{z}_1(0)\mathrm{e}^{-\tau} \geq \lambda\mathrm{e}^{-\tau}$. We obtain that

$$z_1(\tau) \geq \tilde{z}_1(\tau) + y^0_{1,1}(\tau) \geq \lambda,$$

yielding $v_0(t) = 1 - z_1(\tau) \leq 1 - \lambda$.

To establish assertion (ii), consider $\tilde{z}_k(0) = z_k(0)$ for all $k \geq 1$. Then just like in the proof of Lemma 3.3, noting that $A(l, \tau) \leq A(2, \tau) + l - 2$ for all $l \geq 2$,

$$\begin{aligned}\int_{s=0}^{\tau} \tilde{z}_1(s)\mathrm{d}s &= \tilde{Q}(0) - \sum_{l=1}^{\infty} v_l(0)A(l,\tau) \\ &= \sum_{l=1}^{\infty} l v_l(0) - \sum_{l=1}^{\infty} v_l(0)A(l,\tau) = \sum_{l=1}^{\infty} v_l(0)[l - A(l,\tau)] \\ &= \sum_{l=1}^{\infty} v_l(0)[1 - A(1,\tau)] + \sum_{l=2}^{\infty} v_l(0)[l - 1 + A(1,\tau) - A(l,\tau)] \\ &\geq z_1(0)[1 - A(1,\tau)] + \sum_{l=2}^{\infty} v_l(0)[1 + A(1,\tau) - A(2,\tau)] \\ &= z_1(0)[1 - \mathrm{e}^{-\tau}] + z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}].\end{aligned}$$

Also, $t_0 = \tau$, and thus

$$\int_{s=0}^{\tau} y^0_{1,1}(s)\mathrm{d}s = y^0_{0,1}(\tau) = \lambda[\tau - 1 + \mathrm{e}^{-\tau}].$$

We obtain that

$$\begin{aligned}\int_{s=0}^{T} [1 - v_0(s)]\mathrm{d}s = \int_{s=0}^{\tau} z_1(s)\mathrm{d}s &\geq \int_{s=0}^{\tau} [y^0_{1,1}(s) + \tilde{z}_1(s)]\mathrm{d}s \\ &\geq \lambda[\tau - 1 + \mathrm{e}^{-\tau}] + z_1(0)[1 - \mathrm{e}^{-\tau}] + z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}] \\ &= \lambda\tau + [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] + z_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}].\end{aligned}$$

To prove statement (iii), consider as before $\tilde{z}_k(0) = z_k(0)$ for all $k \geq 1$. Further observe that

$$A(l,\tau) \geq A(l-1,\tau) + A(1,\tau),$$

and

$$A(k,\tau) - k = -B(k,\tau) \geq -\tau.$$

Then just like in the proof of Lemma 3.3, noting that $A(l,\tau) \leq A(2,\tau) + l - 2$ for all $l \geq 2$,

$$\begin{aligned}
\tilde{Q}(\tau) =& \sum_{l=1}^{\infty} v_l(0) A(l,\tau) = \sum_{l=1}^{\infty} v_l(0) l + \sum_{l=1}^{\infty} v_l(0)[A(l,\tau) - l] \\
=& Q(0) + \sum_{l=1}^{\infty} v_l(0)[A(1,\tau) - 1] + \sum_{l=2}^{\infty} v_l(0)[A(l,\tau) - l - A(1,\tau) + 1] \\
\geq& Q(0) + z_1(0)[A(1,\tau) - 1] + \sum_{l=2}^{\infty} v_l(0)[A(l-1,\tau) - l + 1] \\
\geq& z_1(0)[1 - \mathrm{e}^{-\tau}] - z_2(0)\tau.
\end{aligned}$$

Also, $t_0 = \tau$, and thus

$$y_{0,1}^0(\tau) = \lambda[\tau - 1 + \mathrm{e}^{-\tau}].$$

We obtain

$$\begin{aligned}
Q(\tau) &\geq y_{0,1}^0(\tau) + \tilde{Q}(\tau) \geq \lambda[\tau - 1 + \mathrm{e}^{-\tau}] + z_1(0)[1 - \mathrm{e}^{-\tau}] - z_2(0)\tau \\
&= \lambda\tau + [z_1(0) - \lambda][1 - \mathrm{e}^{-\tau}] - z_2(0)\tau.
\end{aligned}$$

To establish assertion (iv), consider $\tilde{z}_k(0) = \min\{z_k(0), 1 - \lambda\tau\}$ for all $k \geq 1$. Then, just like in the proof of statement (ii),

$$\begin{aligned}
\int_{s=0}^{\tau} \tilde{z}_1(s)\mathrm{d}s &= \tilde{Q}(0) - \sum_{l=1}^{\infty} \tilde{v}_l(0) A(l,\tau) \\
&= \sum_{l=1}^{\infty} l\tilde{v}_l(0) - \sum_{l=1}^{\infty} \tilde{v}_l(0) A(l,\tau) = \sum_{l=1}^{\infty} \tilde{v}_l(0)[l - A(l,\tau)] \\
&= \sum_{l=1}^{\infty} \tilde{v}_l(0)[1 - A(1,\tau)] + \sum_{l=2}^{\infty} \tilde{v}_l(0)[l - 1 + A(1,\tau) - A(l,\tau)] \\
&\geq \tilde{z}_1(0)[1 - A(1,\tau)] + \sum_{l=2}^{\infty} \tilde{v}_l(0)[1 + A(1,\tau) - A(2,\tau)] \\
&= \tilde{z}_1(0)[1 - \mathrm{e}^{-\tau}] + \tilde{z}_2(0)[1 - \mathrm{e}^{-\tau} - \tau\mathrm{e}^{-\tau}].
\end{aligned}$$

Also, noting that $\tilde{v}_0(0) = 1 - \tilde{z}_1(0) \geq \lambda\tau$ yields $t_0 = \tau$, and thus $y_{1,1}^0(\tau) = \lambda(1 - \mathrm{e}^{-\tau})$.

We obtain that

$$\begin{aligned}\int_{s=0}^{\tau}[1-v_0(s)]\mathrm{d}s &= \int_{s=0}^{\tau} z_1(s)\mathrm{d}s \geq \int_{s=0}^{\tau}[y_{1,1}^0(s)+\tilde{z}_1(s)]\mathrm{d}s\\ &\geq \lambda[\tau-1+\mathrm{e}^{-\tau}]+\tilde{z}_1(0)[1-\mathrm{e}^{-\tau}]+\tilde{z}_2(0)[1-\mathrm{e}^{-\tau}-\tau\mathrm{e}^{-\tau}]\\ &= \lambda\tau+[\tilde{z}_1(0)-\lambda][1-\mathrm{e}^{-\tau}]+\tilde{z}_2(0)[1-\mathrm{e}^{-\tau}-\tau\mathrm{e}^{-\tau}].\end{aligned}$$

Statement (v) follows from statements (ii) and (iv).

3.B Derivation of the fixed point

For convenience, denote by $m^* = \min(j|w_j^* > 0)$ the minimum queue estimate associated with the fixed point. Further denote $n^* = m^*$ if $u_{m^*-1}^* \leq \lambda$, or $n^* = \min\{n : u_n^* > \lambda\}$ otherwise.

Setting the derivatives in (3.11) equal to zero and denoting $\nu = \zeta / w_{n^*}$, we deduce

$$\begin{aligned}0 &= y_{i+1,j}^*\mathbb{1}\{i+1\leq j\}-y_{i,j}^*\mathbb{1}\{i>0\}\\ &\quad+\nu y_{i-1,j-1}^*\mathbb{1}\{n^*=j-1\}-\nu y_{i,j}^*\mathbb{1}\{n^*=j\}\\ &\quad+\delta\sum_{k=0}^{i} v_k^*\mathbb{1}\{i=j=n^*\}+\delta v_i^*\mathbb{1}\{i=j\geq n^*+1\}-\delta y_{i,j}^*\end{aligned}\tag{3.19}$$

for all $i = 0,1,\ldots,j \geq n^*$. Similarly, we have for $j_0 \geq n^*+2$,

$$0 = \frac{d}{dt}\sum_{j=j_0}^{\infty} w_j(t) = \delta\sum_{j=j_0}^{\infty}[v_j^*-w_j^*] = -\delta\sum_{i=0}^{j_0-1}\sum_{j=j_0}^{\infty} y_{i,j}^* \tag{3.20}$$

which yields $y_{i,j}^* = 0$ for all $j \geq n^*+2$ and $i < j$. Additionally, applying (3.19) with $i = k+1$ and $j = k+2$, gives

$$0 = y_{k+2,k+2}^* - y_{k+1,k+2}^* - \delta y_{k+1,k+2}^* = y_{k+2,k+2}^*$$

for all $k \geq n$. In conclusion, it is readily seen that $w_j^* = 0$ for all $j \geq n^*+2$. This implies $m^* = n^*$, and yields

$$y_{i+1,m^*}^*\mathbb{1}\{i\neq m^*\}-(\mathbb{1}\{i\neq 0\}+\nu+\delta)y_{i,m^*}^*+\delta\sum_{k=0}^{m^*} v_k^*\mathbb{1}\{i=m^*\}=0,$$

for all $i = 0, 1, \ldots, m^*$, and

$$\begin{aligned} & y^*_{i+1,m^*+1}\mathbb{1}\{i \neq m^*+1\} - (\mathbb{1}\{i \neq 0\} + \delta) y^*_{i,m^*+1} \\ & + \nu y_{i-1,m^*}\mathbb{1}\{i \neq 0\} + \delta y^*_{m^*+1,m^*+1}\mathbb{1}\{i = m^*+1\} = 0, \end{aligned}$$

for all $i = 0, 1, \ldots, m^*+1$.

We obtain (with $\mu \equiv 1$)

$$\begin{aligned} (\nu+\delta) y^*_{0,m^*} &= \mu y^*_{1,m^*} \\ (\mu+\nu+\delta) y^*_{i,m^*} &= \mu y^*_{i+1,m^*}, \qquad i = 1, \ldots, m^*-1 \\ (\mu+\nu) y^*_{m^*,m^*} &= \delta \left[\sum_{i=0}^{m^*-1} y^*_{i,m^*} + \sum_{i=0}^{m^*} y^*_{i,m^*+1} \right], \end{aligned}$$

or equivalently,

$$(\mu+\nu+\delta) y^*_{m^*,m^*} = \delta \sum_{i=0}^{m^*} [y^*_{i,m^*} + y^*_{i,m^*+1}] = \delta[1 - y^*_{m^*+1,m^*+1}], \tag{3.21}$$

and

$$\begin{aligned} \delta y^*_{0,m^*+1} &= \mu y^*_{1,m^*+1} \\ (\mu+\delta) y^*_{i,m+1} &= \mu y^*_{i+1,m^*+1} + \nu y^*_{i-1,m^*}, \quad i = 1, \ldots, m^* \\ \mu y^*_{m^*+1,m^*+1} &= \nu y^*_{m^*,m^*}, \end{aligned} \tag{3.22}$$

or equivalently,

$$(\mu+\delta) y^*_{m^*+1,m^*+1} = \nu y^*_{m^*,m^*} + \delta y^*_{m^*+1,m^*+1},$$

along with

$$\sum_{i=0}^{m^*} y^*_{i,m^*} + \sum_{i=0}^{m^*+1} y^*_{i,m^*+1} = 1.$$

Note that Equations (3.21) and (3.22) determine $y^*_{m,m}$ and $y^*_{m+1,m+1}$:

$$\begin{aligned} y^*_{m,m} &= \frac{\delta}{(1+\nu)(1+\delta)}, \\ y^*_{m+1,m+1} &= \frac{\nu\delta}{(1+\nu)(1+\delta)}. \end{aligned}$$

It follows from the above equations (flux up equals flux down) that

$$\delta \sum_{i=0}^{m^*-1}(m^*-i)[y^*_{i,m^*}+y^*_{i,m^*+1}]+\nu w^*_{m^*}=\mu\left[\sum_{i=1}^{m^*}y^*_{i,m^*}+\sum_{i=1}^{m^*+1}y^*_{i,m^*+1}\right],$$

which implies that

$$\nu=\frac{\lambda-\Delta}{w^*_m},$$

with

$$\Delta=\delta\sum_{i=0}^{m^*-1}(m^*-i)[y^*_{i,m^*}+y^*_{i,m^*+1}],$$

is equivalent with

$$y^*_{0,m^*}+y^*_{0,m^*+1}=1-\frac{\lambda}{\mu},$$

reflecting that each server is idle a fraction of the time $1-\lambda/\mu$.

Recall that $a=\frac{1}{1+\delta}$ and $b=\frac{1}{1+\delta+\nu}$. We can use the above equations to express y_{m^*-j,m^*} in $y^*_{m^*-j+1,m^*+1}$ for all $j=1,\ldots,m^*$, and recursively obtain

$$\begin{aligned}
y^*_{m^*-j,m^*}&=b^j y^*_{m^*,m^*},\quad j=0,\ldots,m^*-1\\
y^*_{i,m^*}&=b^{m^*-i}y_{m^*,m^*},\quad i=1,\ldots,m^*\\
y^*_{0,m^*}=y^*_{m^*,m^*}&=\frac{b^{m^*-1}}{\nu+\delta}y_{m^*,m^*}.
\end{aligned}$$

Next, we express $y^*_{m^*-j,m^*+1}$ in terms of $y^*_{m^*-j+1,m^*+1}$ and $y^*_{m^*-j-1,m^*}$, and recursively derive

$$\begin{aligned}
y_{m^*+1,m^*+1}=&\nu y_{m^*,m^*}\\
y^*_{m^*-j,m^*+1}=&a^{j+1}y^*_{m^*+1,m^*+1}+\nu ab\sum_{k=0}^{j}a^{j-k}b^k y^*_{m^*,m^*}\\
=&a^{j+1}y^*_{m^*+1,m^*+1}+[a^{j+1}-b^{j+1}]y^*_{m^*,m^*}
\end{aligned}$$

for $j=-1,\ldots,m^*-2$,

$$\begin{aligned}
y^*_{m^*+1-j,m^*+1}=&a^j y^*_{m^*+1,m^*+1}+\nu ab\sum_{k=0}^{j-1}a^{j-k}b^k y^*_{m^*,m^*}\\
=&a^j y^*_{m^*+1,m^*+1}+[a^j-b^j]y^*_{m^*,m^*}
\end{aligned}$$

for $j = 0, \dots, m^* - 1$,

$$\begin{aligned} y^*_{i,m^*+1} &= a^{m^*+1-i} y^*_{m^*+1,m^*+1} + vab \sum_{k=0}^{m^*-i} a^{m^*-i-k} b^k y^*_{m^*,m^*} \\ &= a^{m^*+1-i} y^*_{m^*+1,m^*+1} + [a^{m^*-i+1} - b^{m^*-i+1}] y^*_{m^*,m^*} \end{aligned}$$

for $i = 2, \dots, m^* + 1$,

$$\begin{aligned} y^*_{1,m^*+1} &= a^{m^*} y^*_{m^*+1,m^*+1} + vab \left[\sum_{k=0}^{m^*-2} a^{m^*-1-k} b^k + \frac{1}{v+\delta} b^{m^*-2} \right] y^*_{m^*,m^*} \\ &= a^{m^*} y^*_{m^*+1,m^*+1} + a \left[a^{m^*-1} - \frac{\delta}{v+\delta} b^{m^*-1} \right] y^*_{m^*,m^*} \end{aligned}$$

and $y^*_{0,m^*+1} = \frac{1}{\delta} y^*_{1,m^*+1}$.

It only remains to be shown that Equation (3.15) has a unique solution $v \geq 0$, which then further implies that

$$v = \frac{\lambda - \Delta}{w^*_{m^*}},$$

as noted earlier.

In order to establish that a solution $v \geq 0$ exists, note that $y^*_{0,m^*+1} \downarrow 0$, and

$$y_{0,m^*} \rightarrow \left(\frac{1}{1+\delta} \right)^{m^*} \leq 1 - \lambda,$$

as $v \downarrow 0$, while $y_{0,m^*} \downarrow 0$ and

$$y_{0,m^*+1} \rightarrow \left(\frac{1}{1+\delta} \right)^{m^*+1} > 1 - \lambda$$

as $v \rightarrow \infty$.

It may further be shown that $y_{0,m^*} + y_{0,m^*+1}$ is in fact (strictly) decreasing in v, ensuring that the value of v is also unique.

Chapter 4

Optimal hyper-scalable load balancing with a strict queue limit

Based on:

[BBL20] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. "Optimal Hyper-Scalable Load Balancing with a Strict Queue Limit". In: *Preprint* (2020)

4.1 Introduction

Just as in Chapter 3, we are interested in hyper-scalable load balancing algorithms that are able to achieve great performance while having low communication overhead. In this chapter, we focus on the optimal performance for a potentially scarce communication budget, and our perspective is fundamentally different compared to Chapter 3 in two respects. First of all, we only consider *dispatcher-driven schemes* that have a communication overhead of no more than δ. When $\delta < 1$, we need to resort to hyper-scalable algorithms. Second, jobs may only be admitted when a strict limit K on the queue position of the job can be guaranteed. This queue limit K can have any value and is offered in systems of any size, as opposed to a zero queue length that is only ensured with high probability in a many-server regime. The combination of a low communication budget per job and a strict admission condition is particularly pertinent for high-volume packet processing applications, where zero delay may not be feasible given the admissible message rate, but where an explicit queue limit is crucial.

As the cornerstone of our analysis, we establish a universal upper bound for the achievable throughput of any dispatcher-driven algorithm as function of δ and K, thus capturing the trade-off between performance and communication overhead. We also introduce and analyze a specific hyper-scalable scheme which approaches the latter bound in a many-server regime, demonstrating that the bound is sharp and that the proposed scheme is asymptotically throughput-optimal given the communication and queue limit constraints.

Organization of the chapter. In Section 4.2 we introduce the model and discuss our key findings and contributions. In Section 4.3 we introduce the upper bound for the throughput. The analysis of the hyper-scalable scheme, using a closed queueing network, is presented in Section 4.4. In Section 4.5 we provide simulation results to further illustrate the behavior of the hyper-scalable scheme. An extension of the hyper-scalable scheme that also aims to minimize queue lengths is introduced in Section 4.6. In Section 4.7 we establish product-form distributions for a general closed queueing network scenario which covers both the hyper-scalable scheme and the extension as special cases. We conclude with some remarks and suggestions for further research in Section 4.8.

4.2 Model description and key results

We consider a system with N identical servers of unit exponential rate and a single dispatcher where jobs arrive as a Poisson process of rate $N\lambda$. The dispatcher is unaware of the service requirements of jobs and cannot buffer them, but must immediately forward them to one of the servers or block them. The throughput of the system is defined as the rate of admitted jobs per server.

The blocking option is relevant since the dispatcher must enforce an explicit queue limit K, and is only allowed to admit a job and assign it to a server if it can guarantee that the queue position encountered by that job is at most K. Note that it is not enough for a job to end up in such a position thanks to a lucky guess, but that the dispatcher must have absolute certainty in advance that this is the case, and that a job must be discarded otherwise. Discarding may be the preferred option in packet processing applications when handling a packet beyond a certain tolerance window serves no useful purpose. In that case, processing an obsolete packet results in an unnecessary resource wastage and needlessly contributes to further congestion, and is thus worse than simply dropping the packet upfront.

As mentioned above, the dispatcher is oblivious of the service requirements, which are exponentially distributed and thus have unbounded support. Hence, the dispatcher critically relies on information provided by the servers in order to enforce the queue limit K, and is allowed to send probes for this purpose, requesting queue length reports at a rate $N\delta$. In addition, the dispatcher is endowed with unlimited memory capacity, which it may use to determine which server to probe and when or to which server it will dispatch an arriving job. Servers return instantaneous queue length reports in response to probes from the dispatcher, but are not able to initiate messages or send unsolicited updates when reaching a certain status.

With the above framework in place, we will construct a specific hyper-scalable scheme which is guaranteed to enforce the queue limit K and operate within the communication budget δ. The scheme toggles each individual server between two modes of operation, labeled open and closed. An open period starts when the dispatcher requests a queue length update from the server and the reported queue length is below K; during that period the server is not working, and waits for incoming jobs from the dispatcher, seeing its queue only grow. Once the queue length reaches the limit K, a closed period starts, ending when the dispatcher requests the next update after τ time units; during that period the server is continuously working as long as jobs are available, without receiving any further jobs, thus draining its queue. When the queue length reported at an update is exactly K, the open period has length zero, and the next closed period starts immediately. By construction, the above-described mechanism maintains a queue limit of K at all times and induces a message rate of at most $1/\tau$ per server, which makes $\tau = 1/\delta$ the obvious choice.

The scheme is similar to AUJSQ$^{\text{det}}(\delta)$ from Chapter 3. The most important difference is that in AUJSQ$^{\text{det}}(\delta)$, every server is updated *exactly* every $\tau = 1/\delta$ time units based on a timer, while the current scheme only starts the timer when the queue length of a server has reached K. Thus the AUJSQ(δ) scheme might update servers even when they are known to have strictly less than K jobs in queue. This difference however typically vanishes when the number of servers increases and the load approaches the critical value $\lambda^*(\delta, K)$. A more detailed comparison accompanied with simulation results will be provided in Section 4.5.4.

Main results. We now discuss the main results, which can be summarized as follows. There is a function λ^* of δ and K, such that subject to a message rate δ and queue limit K,

- the throughput of any dispatcher-driven algorithm is bounded from above by $\min\{\lambda^*(\delta, K), \lambda\}$,
- the throughput of our hyper-scalable scheme approaches $\min\{\lambda^*(\delta, K), \lambda\}$ as $N \to \infty$.

These two results are covered in Sections 4.3 and 4.4, respectively.

4.3 Universal upper bound

We establish the upper bound for a slightly more general scenario with heterogeneous server speeds. Denote the speed of the n-th server by μ_n for $n = 1, \ldots, N$. The next theorem shows that the achievable throughput of any dispatcher-driven algorithm subject to the message rate δ and queue limit K is bounded from above by

$$\lambda^*(\delta, K) = \delta B(K, \bar{\mu}/\delta), \tag{4.1}$$

with

$$B(K, \tau) = \sum_{k=0}^{K-1} \left(1 - \mathrm{e}^{-\tau} \sum_{i=0}^{k} \frac{\tau^i}{i!} \right), \tag{4.2}$$

and $\bar{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mu_n$ denoting the system-wide average server speed. Note that $B(K, \tau)$ may be equivalently written as

$$B(K, \tau) = K - \sum_{k=0}^{K-1} (K - k) \mathrm{e}^{-\tau} \frac{\tau^k}{k!},$$

and may be interpreted as the expected value of the minimum of K and a Poisson distributed random variable with mean τ, just as in Section 3.3.2.

Theorem 4.1. *The expected number of jobs that any dispatcher-driven algorithm can admit subject to the queue limit K during a period of length T_0 with at most $\delta N T_0$ message exchanges cannot exceed $2KN + \lambda^*(\delta, K) \times N T_0$, for any $\delta > 0$. In particular, the achievable throughput with a message rate of at most $\delta > 0$ is bounded from above by $\lambda^*(\delta, K)$.*

Recall that we defined throughput as the rate of admitted jobs per server, and note that the throughput is naturally bounded from above by the normalized arrival rate λ.

Proof. As noted earlier, since the execution times are exponentially distributed and thus have unbounded support, the dispatcher relies on information provided by the servers in order to enforce the queue limit K. Specifically, the dispatcher earns 'passes' for admitting k jobs when a server reports $k = 0, \ldots, K$ service completions since the previous update, and cannot admit any job without relinquishing a pass. Thus, the number of jobs that the dispatcher can admit during a particular time period cannot exceed the sum of (i) the maximum possible number of KN passes initially available; (ii) the maximum possible number of KN passes earned at the first update from each server during that period, if any; and (iii) the number of additional passes obtained at further updates over intervals that fell entirely during that period, if any. Now suppose that the dispatcher requests L_n queue length reports from the n-th server during a period of length T_0, one after each of the update intervals of lengths $T_{n,1}, \ldots, T_{n,L_n}$, with $\sum_{l=1}^{L_n} T_{n,l} \leq T_0$ for all $n = 1, \ldots, N$ and $L = \sum_{n=1}^{N} L_n \leq \delta N T_0$. Then the number of passes earned at the l-th update equals the number of service completions during the time interval $T_{n,l}$, which depends on the queue length at the start of that interval. However, this random variable is stochastically bounded from above by when the queue was full with K jobs at the start of the interval. In the latter case the number of passes earned is given by the minimum of K and a Poisson distributed random variable with parameter $\mu_n T_{n,l}$. We deduce that the expected total number of passes obtained at all these updates is bounded from above by

$$\sum_{n=1}^{N} \sum_{l=1}^{L_n} B(K, \mu_n T_{n,l}), \tag{4.3}$$

and to prove the first statement of the theorem it thus remains to be shown that this quantity is no larger than $\lambda^*(\delta, K) \times N T_0$. It is easily verified that

$$\frac{\partial^2 B(K,t)}{\partial t^2} = -\mathrm{e}^{-t} \frac{t^{K-1}}{(K-1)!} < 0,$$

implying that $B(K,t)$ is concave as function of t. As an aside, the above expression may be intuitively explained from the fact that the first derivative $\frac{\partial B(K,t)}{\partial t}$ equals the probability that exactly $K-1$ unit-rate Poisson events occur during a period of length t, while the (negative) derivative of the latter probability equals that very same probability by virtue of the Kolmogorov equations for a pure birth process. Because of concavity, we obtain that (4.3) is no larger than $L \times B(K, \tau)$, with

$$\tau = \frac{1}{L} \sum_{n=1}^{N} \mu_n \sum_{l=1}^{L_n} T_{n,l} \leq \frac{1}{L} \sum_{n=1}^{N} \mu_n T_0 = \bar{\mu} \frac{N T_0}{L}. \tag{4.4}$$

Invoking the fact that $\frac{\partial B(K,t)}{\partial t} > 0$, i.e., $B(K, t)$ is increasing in t, we may write

$$L \times B(K, \tau) \leq L \times B(K, \bar{\mu} N T_0 / L) = \lambda^*(\gamma, K) \times N T_0, \tag{4.5}$$

with $\gamma = \frac{L}{N T_0} \leq \delta$. It is easily verified that

$$\frac{\partial \lambda^*(x, K)}{\partial x} = K - K \mathrm{e}^{-1/x} \sum_{k=0}^{K} \frac{(1/x)^k}{k!} > 0, \tag{4.6}$$

i.e., $\lambda^*(x, K)$ is increasing in x, and hence $\lambda^*(\gamma, K) \leq \lambda^*(\delta, K)$, which completes the proof of the first statement of the theorem.

Finally, to prove the second statement, we consider the long-term scenario $T_0 \to \infty$. The number of jobs that are admitted per time-unit per server then equals $(2KN + \lambda^*(\delta, K) \times N T_0)/(N T_0) \to \lambda^*(\delta, K)$ and the message rate per server equals at most $\delta N T_0 / (N T_0) = \delta$. □

Properties of λ^*. We now state some properties of $\lambda^*(\delta, K)$ and discuss their consequences, where we assume without loss of generality that $\bar{\mu} = 1$. In the next subsection we will introduce a hyper-scalable scheme which is able to achieve this throughput in the many-server regime. For now, we will reflect the properties in light of the maximum throughput that is possible for any dispatcher-driven load balancing algorithm given a message rate δ.

Proposition 4.1. *$\lambda^*(\delta, K)$ has the following properties:*

(i) $\lambda^*(\delta, K)$ is strictly increasing in both δ and K,

(ii) $\lambda^*(\delta, K) \uparrow 1$ as $\delta \to \infty$,

(iii) $\lambda^*(\delta, K) \downarrow 0$ and $\lambda^*(\delta, K)/\delta \to K$ as $\delta \downarrow 0$,

(iv) for $a \leq 1$, $\lambda^*(a/K, K) \to a$ as $K \to \infty$.

Proof. $\lambda^*(\delta, K)$ is strictly increasing in δ because of (4.6) and is strictly increasing in K since $1 - e^{-\tau}\sum_{i=0}^{k}\frac{\tau^i}{i!} > 0$ in (4.2). For Properties (ii) to (iv), note that

$$\begin{aligned}&\delta[K - Ke^{-1/\delta} - e^{-1/\delta}((K-1)/\delta + (K-2)(1/\delta)^{y(\delta)})]\\&\le \delta(K - e^{-1/\delta}\sum_{i=0}^{K-1}(K-i)\frac{(1/\delta)^i}{i!}) = \lambda^*(\delta, K) \le \min(\delta K, 1),\end{aligned}$$

with $y(\delta) = 2$ when $\delta \ge 1$ and $y(\delta) = K - 1$ when $\delta < 1$. All limiting statements are true for the LHS and RHS of the previous equation, therefore proving these properties for $\lambda^*(\delta, K)$ too. $\square$

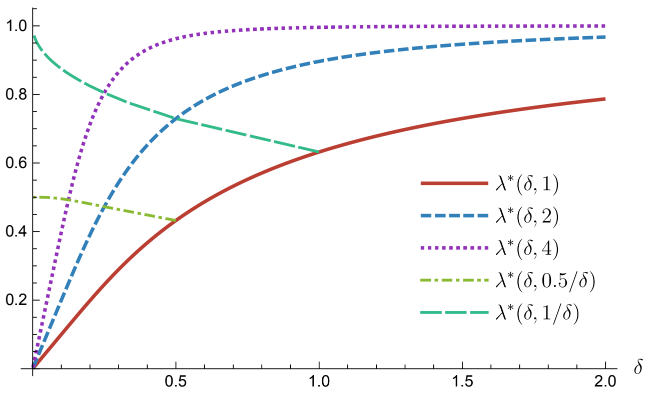


Figure 4.1: Visualization of the throughput bound $\lambda^*(\delta, K)$ for various values of K as function of δ. For the fourth and fifth graph, only values of δ are evaluated for which the second argument is integer-valued.

The properties in Proposition 4.1 are visualized in Figure 4.1. They can be interpreted intuitively and practically too. For Property (i), when the communication budget is expanded, i.e. δ is increased, more jobs can be dispatched to queues that are guaranteed to be short. Similarly, more jobs can be admitted into the system if the queue limit is raised, i.e., K is increased. Property (i), in conjunction with Theorem 4.1, implies that a throughput $\lambda^*(\delta, K)$ cannot be achieved with a message rate strictly below δ, or a queue limit strictly below K.

Property (ii) shows that as the message rate grows large, full server utilization can be achieved. With an unlimited message rate, the dispatcher is able to find idle servers immediately, a necessary requirement for achieving full server utilization irrespective of the queue limit K.

Property (iii) shows that, first, when no communication is allowed, no jobs can be sent to queues that are guaranteed to be short. The further specification of the limit indicates that K jobs are admitted into the system per message. This in turn reveals that when the communication is extremely infrequent, all messages result into finding an idle server, and thus provide the dispatcher with K passes to admit jobs.

Finally, Property (iv) is somewhat similar to Property (iii). When the queue limit K increases, one needs fewer messages in order to achieve a server utilization level a. With $a = 1$, Property (iv) shows that one message per K jobs is needed in order to achieve full server utilization, which is a somewhat similar conclusion as the one from Property (iii).

4.4 The hyper-scalable scheme

We now introduce the hyper-scalable scheme in full detail for the case of homogeneous servers.

At all times, the dispatcher remembers the most recent queue length that was reported by every server. Furthermore, the dispatcher records the number of jobs that have been sent to every server since the last update from that server. When the sum of these two numbers is strictly less than the queue limit K, a server is labeled open, and otherwise closed.

Whenever a job arrives to the dispatcher, it is assigned to an open server, if possible. There are two options for how to select an open server. Either an open server is selected uniformly at random (random case), or the open server that was interacted with (i.e. updated or received job) the longest ago is selected (FCFS case). The job is dropped when no open servers exist.

Exactly τ time units after a server was labeled closed, the dispatcher will request a queue length update of the server. The server becomes open when this queue length is strictly less than K, and the server remains closed for another τ time units if the queue length equals K, in which case the dispatcher will request the next queue length update after another τ time units. The hyper-scalable scheme is a dispatcher-driven algorithm, since only the dispatcher initiates messages and every server can track itself when it is labeled open by the dispatcher: exactly when the sum of the queue length during the latest update and the number of jobs received since then is strictly below K.

Note that by construction the hyper-scalable scheme respects the queue limit K at all times and involves a message rate of at most $1/\tau$. In addition, the scheme has been specifically designed to allow explicit analysis and derivation of provable capacity benchmarks. As it turns out, a crucial feature in that regard is for the servers to refrain from executing jobs while being marked open. This feature ensures that the queue length is exactly K at the moment a server becomes closed. The average number of job completions in an interval of length τ then equals $B(K,\tau)$, so one message leads to $B(K,\tau)$ admitted jobs on average, immediately yielding the following result.

Corollary 4.1. *The average number of messages per admitted job equals* $1/B(K,\tau)$*, regardless of* λ *and* N*.*

While the forced idling of servers during open periods may seem inefficient, the next theorem shows that the proposed hyper-scalable scheme is in fact throughput-optimal in large-scale systems, given the message rate δ and queue limit K, with the choice $\tau = 1/\delta$.

Theorem 4.2. *For any* $\delta > 0$*, the throughput that is achieved by the hyper-scalable scheme with* $\tau = 1/\delta$ *approaches* $\min\{\lambda^*(\delta,K),\lambda\}$ *as* $N \to \infty$*.*

Since the hyper-scalable scheme obeys the queue limit K and involves a message rate of at most δ, Theorems 4.1 and 4.2 combined imply that it is throughput-optimal as $N \to \infty$.

According to Theorem 4.1 and Property (i) of Proposition 4.1, one would require a message rate of at least δ to achieve a throughput of $\lambda^*(\delta,K)$. Theorem 4.2 shows that the throughput of the hyper-scalable scheme approaches $\lambda^*(\delta,K)$ as $N \to \infty$ when $\lambda \geq \lambda^*(\delta,K)$. A combination of these two observations (and the fact that $\lambda^*(\delta,K)$ is continuous in δ) indicates that the message rate of the hyper-scalable scheme must approach δ as $N \to \infty$ when $\lambda \geq \lambda^*(\delta,K)$. This in turn implies that the expected duration of an open period must become negligible, compare to the length τ of a closed period, i.e. the fraction of time that a server is marked open vanishes.

We now proceed with an outline of the proof of Theorem 2.

Analysis. For brevity, a server is said to be in state k when the sum of the queue length at its latest update epoch and the number of jobs the server has received since, equals k. This means that all servers in state $k < K$ are labeled open and servers in state K are labeled closed. In view of the homogeneity of the servers, it is useful to further introduce $N(t) = (N_0(t), N_1(t), \dots, N_{K-1}(t), N_K(t))$, with $\sum_k N_k(t) = N$, where $N_k(t)$ stands for the number of servers in state k at time t. While the vector $N(t)$ provides a convenient representation, it is worth emphasizing that it does not provide a Markovian state description.

We now explain how individual servers transition between various states. When a job arrives to the system, the state of an open server will change from $k < K$ to $k+1$. An update of a server may cause the server to change state too. The new state of the server equals the number of jobs that are left in queue after the update interval of τ time units. The number of jobs that were served follows a truncated Poisson distribution, so the probability p_k that exactly k jobs remain, equals $p_k := \mathrm{e}^{-\tau} \frac{\tau^{K-k}}{(K-k)!}$ for $k > 0$ and $p_0 := 1 - \mathrm{e}^{-\tau} \sum_{i=0}^{K-1} \frac{\tau^i}{i!}$. When $k < K$ jobs are left, the state of the server becomes k. When there are K jobs left, the state of the server does not change and remains K.

It is important to observe that service completions of jobs do not cause direct transitions in server states. The reason is twofold. When a server is open, it stops working on jobs, so there are no such completions at open servers. For closed servers, all servers are aggregated; the number of jobs in queue is not taken into account. Only after the period of length τ, the number of jobs left in queue is determined indirectly by using the transition probabilities as specified above.

Although the vector $N(t)$ does not provide a Markovian state description as noted above, its evolution can be described in terms of a closed queueing network, in which the servers act as customers in the network, traversing various nodes corresponding to their states. Specifically, the closed queueing network consists of one multi-class "single-server" node with service rate λN in which the customers can be of classes $0, 1, \dots, K-1$, and one "infinite-server" node with deterministic service time τ that holds all class-K customers. A service completion at the single-server node makes one customer transition. The class of the customer changes from k to $k+1$ if $k < K-1$, or the customer transitions to the infinite-server node if its class was $K-1$. When multiple customers are present at the single-server node, the customer that transitions is either selected uniformly at random (random case), or the customer that has been in the single-server node for the longest time is selected (FCFS case). Finally, upon a service completion at the infinite-server node a customer moves to the single-server node as class $k < K$ with probability p_k, or directly re-enters the infinite-server node with probability p_K.

A schematic representation is shown in Figure 4.2. We define γ_k as the relative throughput value of class-k customers. With $\gamma_K = 1$, it follows that $\gamma_k = p_0 + \ldots + p_k = 1 - \mathrm{e}^{-\tau} \sum_{i=0}^{K-1-k} \frac{\tau^i}{i!}$ for $k < K$.

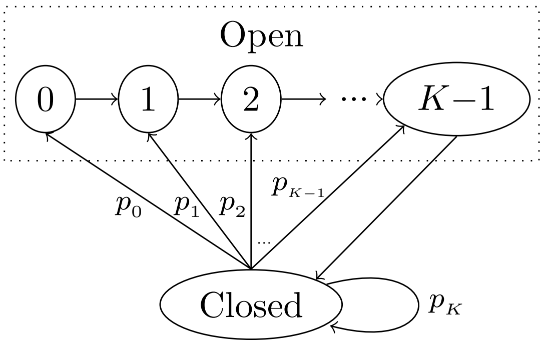


Figure 4.2: Schematic representation of the circulation of an individual customer in the closed queueing network.

By virtue of the above-described equivalence, the process $N(t)$ representing the server states under the hyper-scalable scheme inherits the product-form equilibrium distribution of the closed network as stated in the next proposition.

Proposition 4.2. *The equilibrium distribution of the system with N servers is*

$$\pi(n_0, n_1, \ldots, n_{K-1}, n_K) = G_N^{-1} \frac{(n_0 + \ldots + n_{K-1})!}{n_0! \ldots n_{K-1}!} \left(\prod_{i=0}^{K-1} \left(\frac{\gamma_i}{\lambda N} \right)^{n_i} \right) \frac{\tau^{n_K}}{n_K!} \tag{4.7}$$

if $n_0 + \ldots + n_K = N$, with normalization constant

$$G_N = \sum_{v_0 + \ldots + v_{K-1} + w = N} \frac{(v_0 + \ldots + v_{K-1})!}{v_0! \ldots v_{K-1}!} \left(\prod_{i=0}^{K-1} \left(\frac{\gamma_i}{\lambda N} \right)^{v_i} \right) \frac{\tau^w}{w!}.$$

A proof of Proposition 4.2 can be found in Section 4.7, and the product-form equilibrium distribution may be informally understood as follows. The infinite-server node allows a product-form distribution even for deterministic service times. While traditionally exponentially distributed service times are considered, the equilibrium distribution is insensitive to the service time distribution at the infinite-server node and only depends on its mean, see Section 4.7 for details. As mentioned above, the service discipline at the single-server node with exponentially distributed service times may either be FCFS or random order of service. In the case of the FCFS discipline, albeit not being reversible [Kel11], the single-server node with multiple classes can be represented as an order-independent queue [BKK95; Krz11]. According to Theorem 2.2 in [Krz11], the queue is quasi-reversible, which is sufficient for a product-form distribution. For random order of service, which is a symmetric service discipline, the single-server node is reversible, yielding a product-form as well.

The equilibrium distribution (4.7) can be simplified when only the number of open and closed servers matters. This immediately yields an expression for the blocking probability L_N as provided in the next corollary.

Corollary 4.2. *The equilibrium probability of there being n open servers and $N-n$ closed servers under the hyper-scalable scheme equals*

$$\pi(n, N-n) = \sum_{n_0+\ldots+n_{K-1}=n} \pi(n_0, \ldots, n_{K-1}, N-n) = \frac{\left(\frac{B(K,\tau)}{\lambda N}\right)^n \frac{\tau^{N-n}}{(N-n)!}}{\sum_{w=0}^{N} \left(\frac{B(K,\tau)}{\lambda N}\right)^w \frac{\tau^{N-w}}{(N-w)!}}. \quad (4.8)$$

In particular, because of the PASTA property, the blocking probability is given by

$$L_N = \pi(0, N) = \frac{\frac{(xN)^N}{N!}}{\sum_{w=0}^{N} \frac{(xN)^w}{w!}}, \quad (4.9)$$

with $x = \lambda\tau / B(K,\tau) = \lambda/\lambda^*(1/\tau, K)$. Finally,

$$L_N \overset{N\to\infty}{\to} \max\{0, 1 - \lambda^*(1/\tau, K)/\lambda\}.$$

Specifically, $L_N \downarrow 0$ as $N \to \infty$ when $\lambda \le \lambda^*(1/\tau, K)$.

Suppose that the allowed message rate is δ as stated in Theorem 4.2, then put $\tau = 1/\delta$. When $\lambda \le \lambda^*(1/\tau, K)$, the blocking probability vanishes in the many-server regime according to Corollary 4.2, and thus the throughput approaches λ. When $\lambda > \lambda^*(1/\tau, K)$, the acceptance probability tends to $\lambda^*(1/\tau, K)/\lambda$ and the throughput approaches $\lambda \times \lambda^*(1/\tau, K)/\lambda = \lambda^*(1/\tau, K)$. These two statements combined yield Theorem 4.2.

Theorem 4.2 allows us to equivalently view $\lambda^*(\delta, K)$ as the throughput that is achieved by the hyper-scalable scheme as $N \to \infty$ when $\lambda \ge \lambda^*(\delta, K)$. We now revisit properties (ii) and (iii) as stated in Proposition 4.1 from that perspective. In the limiting scenario $\delta \to \infty$, $\tau \downarrow 0$, servers are updated after an infinitesimally small time, which in turn alerts the dispatcher immediately when even a single job has been served. This ensures that all servers can work at full capacity.

In the scenario $\delta \downarrow 0$, $\tau \to \infty$, update periods become extremely long. Every update that does happen, will definitely find an idle server and allow for K admitted jobs, explaining why $\lambda^*(\delta, K) \approx K\delta$ for small δ.

Remark. Note that with the queue limit K in force we may assume each server to have a finite buffer of size K. In case of a finite buffer, the queue limit K would automatically be enforced, even if the dispatcher were allowed to forward jobs without any advance guarantee. With the option of "(semi)-blind guesses", the throughput bound would trivially become 1 (the average server speed), and Property (iii) indicates that the achievable throughput $\lambda^*(\delta, K)$ without lucky guesses could be (substantially) lower when δ is (significantly) smaller than $1/K$. However the throughput of 1 can only be approached for a high arrival rate, at the expense of severe blocking, whereas the hyper-scalable scheme can deliver the throughput $\lambda^*(\delta, K)$ with negligible blocking asymptotically.

4.5 Simulation experiments and optimality benchmarks

In this section we conduct various simulation experiments to further benchmark the properties of the hyper-scalable scheme and make several comparisons. Throughout we often set the queue limit $K = 2$, yielding the throughput bound $\lambda^*(\delta, 2) = 2\delta - 2\delta e^{-1/\delta} - e^{-1/\delta}$ as function of the message rate δ. Furthermore, all simulation results emulate the random case, i.e. a job is sent to an open server selected uniformly at random.

4.5.1 Baseline version of the hyper-scalable scheme

First, we evaluate the hyper-scalable scheme itself in Figures 4.3a and 4.3b for $K = 2$ and $K = 3$ respectively. We note that the message rate stays below the line $y = 1/\tau$, confirming that it never exceeds $1/\tau$. The throughput and blocking probability achieved by the hyper-scalable scheme are nearly indistinguishable from the respective asymptotic values (upper and lower bounds, respectively), especially at lower and medium values of the communication budget $1/\tau$. For higher values of the communication budget, the throughput and blocking probability slightly diverge from the asymptotic values but remain remarkably close nevertheless. This demonstrates that the asymptotic optimality properties of

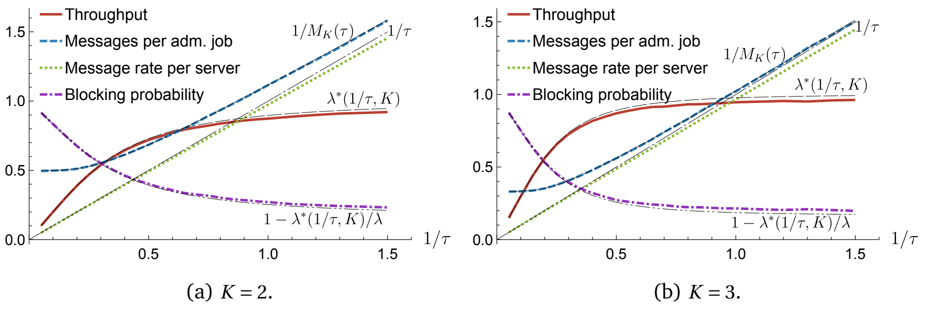


Figure 4.3: Simulation results for the hyper-scalable scheme for $\lambda = 1.2$ and $N = 100$. Numerical values of the throughput bound $\lambda^*(1/\tau, K)$, the associated blocking probability bound $1-\lambda^*(1/\tau, K)/\lambda$, the average number of messages per admitted job $1/B(K,\tau)$ and $1/\tau$ are also shown with thin black lines.

the hyper-scalable scheme as stated in Theorems 4.1 and 4.2 already manifest themselves in moderately large systems.

In order to provide further insight in the asymptotic optimality, we compare the baseline version of the hyper-scalable scheme with several variants and alternative scenarios that are not analytically tractable.

Specifically, in the next two subsections, we examine the following variants through simulations:

- “non-idling”; open servers continue working, but will convey their queue length as if they had not been working while being open,

- “work-conserving”; open servers continue working and convey their actual queue lengths at update epochs.

At first sight, one might suspect that these variants achieve a possibly larger throughput. As we will see however, the differences are small and are only observable at low load or in systems with few servers.

In Section 4.5.4 we make a comparison with the AUJSQ$^{\text{det}}(\delta)$ scheme considered in Chapter 3, which is not analytically tractable either but seems to be asymptotically throughput-optimal as well.

4.5.2 Non-idling variant

Open servers do not work on jobs in the baseline version of the hyper-scalable scheme. While Theorem 4.2 showed that the forced idling does not affect the achieved throughput in large-scale systems, it is still interesting to investigate the consequences of this design. In the non-idling variant, open servers do work on jobs, but they convey their queue length to the dispatcher as if they had not been working on jobs while being labeled open. While this variant may seem fundamentally different, the information that the dispatcher has is exactly the same as in the baseline version: the sets of open servers and their respective states coincide in both scenarios, as long as jobs are sent to the same open server.

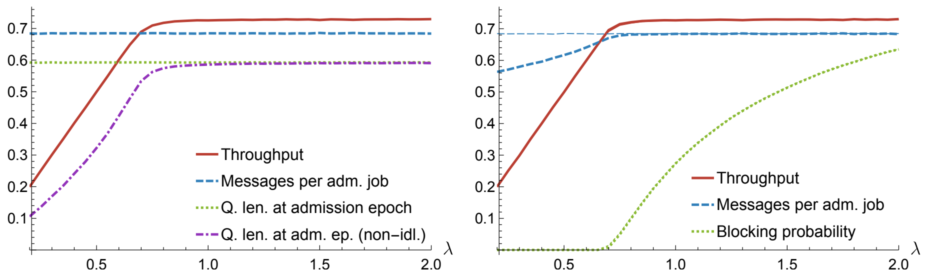
In particular, the equilibrium distribution of the server states as provided in Proposition 4.2 applies to the non-idling variant as well, and the throughput and the number of messages exchanged per admitted job are identical in both scenarios. The only difference arises in the expected queue lengths encountered by admitted jobs: they are somewhat smaller in the non-idling scenario, as illustrated by the simulation results presented in Figure 4.4a.

At low load values, there are instants where there is time for servers to execute jobs when they are open. This causes a distinction between the two variants, since in the non-idling variant jobs join shorter queues. In Section 4.6, we will consider a tractable extension of the hyper-scalable scheme that aims to reduce the queue lengths. As the number of servers grows however, an overflow of arrivals will cause open servers to have less time to execute jobs, which causes the queue lengths to be similar in both scenarios. This viewpoint provides further intuition why the hyper-scalable scheme is still asymptotically optimal.

4.5.3 Work-conserving variant

We now turn to a work-conserving variant of the hyper-scalable scheme, in which open servers also work on jobs, and in fact convey their actual queue length at an update epoch. In this case the evolution of the server states is different, and the equilibrium distribution provided in Proposition 4.2 no longer applies.

The throughput and blocking probability are similar in both scenarios. This may be intuitively explained as follows. When $\lambda \geq \lambda^*(1/2, 2)$, Theorem 4.2 shows that there are hardly ever any open servers, and hence there should not be any substantial difference between the two variants, which is corroborated by Figure



(a) Non-idling variant. (b) Work-conserving variant.

Figure 4.4: Simulation results: comparison between two variants (thick lines) and the baseline scenario (thin lines), for $K = 2$, $\tau = 2$ and $N = 500$, yielding a throughput bound $\lambda^*(1/2, 2) \approx 0.73$.

4.4b.

When $\lambda < \lambda^*(1/2, 2)$, there can be a significant number of open servers. Theorem 4.2 however implies that the hyper-scalable scheme approaches zero blocking and throughput λ in this case. While it is plausible that the work-conserving variant achieves that as well, as attested by Figure 4.4b, it is simply not feasible to achieve lower blocking or higher throughput. The only room for improvement is thus in the number of message exchanges per admitted job, and Figure 4.4b demonstrates that the work-conserving variant indeed provides some gain compared to the hyper-scalable scheme in that regard. To put that observation in perspective, consider Corollary 4.1. As one can see, the communication overhead is strictly decreasing in τ. For such a low arrival rate, the hyper-scalable scheme permits to choose the update interval τ much larger. Figure 4.5 confirms that the choice $\tau = 5$ largely eliminates the difference in communication overhead between the work-conserving variant and the baseline version.

4.5.4 Comparison with the AUJSQ$^{\text{det}}(\delta)$ scheme

We now compare the hyper-scalable scheme with the AUJSQ$^{\text{det}}(\delta)$ scheme from Chapter 3, which is somewhat similar, except that every server is updated *exactly* every $\tau = 1/\delta$ time units based on a timer. Thus the AUJSQ$^{\text{det}}(\delta)$ scheme might update servers even when they are known to have strictly less than $K = 2$ jobs in queue. There are further minor differences: in AUJSQ$^{\text{det}}(\delta)$ jobs are assigned to the server with the lowest state (so giving preference to servers that are more

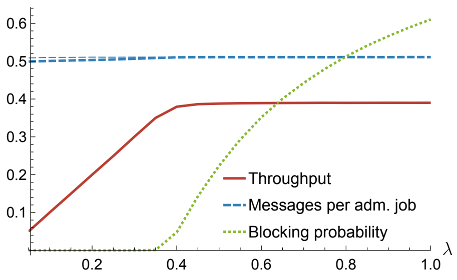


Figure 4.5: Simulation results: comparison between the baseline scenario (thin lines) and the work-conserving variant (thick lines) for $K = 2$, $\tau = 5$ and $N = 500$, so that $\lambda^*(1/5,2) \approx 0.39$.

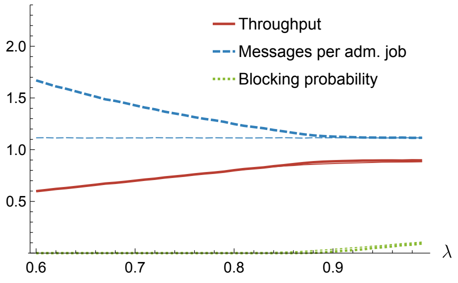


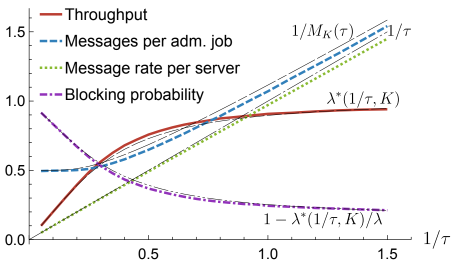
Figure 4.6: Simulation results: comparison between the baseline scenario (thin lines) and the AUJSQ$^{\text{det}}(\delta)$ scheme (thick lines) for $K = 2$, $\tau = 1$ and $N = 500$, so that $\lambda^*(1,2) \approx 0.90$.

likely to be empty) and open servers do work on jobs. In contrast to Chapter 3, we now consider a variant of the AUJSQ$^{\text{det}}(\delta)$ scheme in which jobs are blocked when the dispatcher is not aware of any servers that are guaranteed to have strictly less than $K = 2$ jobs in queue. The comparison is shown in Figure 4.6.

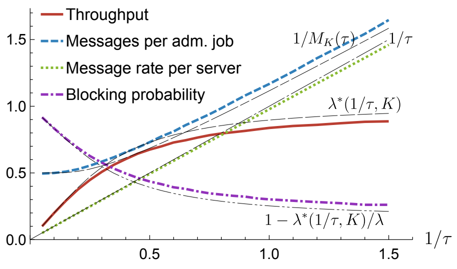
It is important to note that in the hyper-scalable scheme the expected number of messages *per admitted job* is independent of λ, while in the AUJSQ$^{\text{det}}(\delta)$ scheme the expected number of messages *per time unit* is independent of λ. We observe that the average number of messages per admitted job coincides when $\lambda > \lambda^*(1/\tau, K)$. While it is natural to expect that the AUJSQ$^{\text{det}}(\delta)$ scheme offers similar asymptotic optimality properties, it lacks the mathematical tractability of the hyper-scalable scheme to facilitate a rigorous proof argument.

4.5.5 Non-exponential service times

Finally, we analyze the hyper-scalable scheme for non-exponential service time distributions. In Figure 4.7a, the service times are Gamma(2,2) distributed. The throughput of the hyper-scalable algorithm slightly exceeds the value of $\lambda^*(1/\tau, K)$, the maximum throughput when job sizes are exponential. The number of messages per admitted job is also lower than $1/B(K,\tau)$. This is all explained by the fact that the tail of the Gamma(2,2) distribution is smaller than the tail of the exponential distribution. This means that more jobs are completed in a fixed time interval, which increases the effectiveness of the messages sent.



(a) Decreasing hazard rate Gamma(2, 2).



(b) Increasing hazard rate Gamma(1/2, 1/2).

Figure 4.7: Simulation results for the hyper-scalable scheme for $K = 2$, $\lambda = 1.2$ and $N = 100$, and non-exponential service times. Numerical values of the throughput bound $\lambda^*(1/\tau, K)$, the associated blocking probability bound $1-\lambda^*(1/\tau, K)/\lambda$, the average number of messages per admitted job $1/B(K,\tau)$ and $1/\tau$ are also shown with thin black lines.

The service time distribution in Figure 4.7b is Gamma(1/2,1/2). The opposite effect is observed: the throughput is lower compared to the exponential case (Figure 4.3a) and the message rate is larger, because of the heavier tail.

4.6 Extension aimed at minimizing queue lengths

While the hyper-scalable scheme is asymptotically throughput-optimal given the message rate δ and queue limit K, it does not make any explicit effort beyond that to minimize queue lengths or delays experienced by jobs. Motivated by that observation, we now consider an extension of the hyper-scalable scheme aimed at minimizing waiting times. In this extension, a server that receives its i-th job after an update at which its queue length was k, becomes closed for $\tau_{k,k+i}$ time units. After this time, it becomes open if $k+i < K$ and it is updated if $k+i = K$. Thus, servers are not only closed when they become full, but are closed for a while after every job they receive.

Henceforth, we focus on the case $K = 2$ for the ease of exposition, and we set $\tau_{0,0} = 0$, $\tau_{0,1} = \tau_{1,1} = \tau_1$, $\tau_{0,2} = \tau_{1,2} = \tau_2$ and $\tau_{2,2} = \tau_3$. We can put $\tau_{0,0}$ to zero without loss of generality as it makes no sense to have a cool-down period for an empty server. As a consequence there is no difference between servers that had zero jobs or one job at the previous update epoch, so we can set $\tau_{0,1} = \tau_{1,1}$, and $\tau_{0,2} = \tau_{1,2}$ as well. Let p_{2j} be the probability that j jobs remain after an update, when there were zero or one jobs just after the latest update epoch.

This means that the server had $\tau_{0,1}$ time units to work on the first job and another $\tau_{0,2}$ time units after both jobs were dispatched to it. This gives $p_{20} = \mathrm{e}^{-\tau_1}(1-\tau_2\mathrm{e}^{-\tau_2}-\mathrm{e}^{-\tau_2})+(1-\mathrm{e}^{-\tau_1})(1-\mathrm{e}^{-\tau_2})$, $p_{22} = \mathrm{e}^{-\tau_1}\mathrm{e}^{-\tau_2}$ and $p_{21} = 1-p_{20}-p_{22}$. Let q_{2j} be the probability that j jobs remain after an update, when there were two jobs just after the latest update epoch. This gives $q_{20} = 1-\mathrm{e}^{-\tau_3}-\tau_3\mathrm{e}^{-\tau_3}$, $q_{22} = \mathrm{e}^{-\tau_3}$ and $q_{21} = 1-q_{20}-q_{22}$.

Servers can be in either of the five following states.

A_1 The server is idle and open.

B_1 The server had zero jobs during the previous update moment and received one job since, or the server had one job during the previous update moment and received no jobs since. The server is now marked closed for τ_1 time units.

A_2 The server had zero jobs during the previous update moment and received one job since, or the server had one job during the previous update moment and received no jobs since. The server was marked closed for τ_1 but is now open.

B_2 The server had zero jobs during the previous update moment and received two jobs since, or the server had one job during the previous update moment and received one job since. The server is now marked closed for τ_2 time units.

B_3 The server had two jobs during the previous update moment and is now marked closed for τ_3 time units.

The transitions are schematically represented in Figure 4.8, with the transition probabilities as defined earlier.

The system dynamics under this extension of the hyper-scalable scheme can also be represented in terms of a closed queueing network with one single-server node that holds two classes of customers and three infinite-server nodes. The states A_1 and A_2 correspond to the two classes that customers can be of when they are present at the single-server node. The states B_1, B_2 and B_3 each correspond to one of the three infinite-server nodes in the network, with deterministic service times τ_1, τ_2 and τ_3, respectively.

Proposition 4.3. *The equilibrium distribution of the system with N servers is*

$$\pi(n_1,n_2,m_1,m_2,m_3) = H_N^{-1}\frac{(n_1+n_2)!}{n_1!n_2!}\left(\frac{\gamma_1}{\lambda N}\right)^{n_1}\left(\frac{\gamma_2}{\lambda N}\right)^{n_2}\frac{(\kappa_1\tau_1)^{m_1}}{m_1!}\frac{(\kappa_2\tau_2)^{m_2}}{m_2!}\frac{(\kappa_3\tau_3)^{m_3}}{m_3!}$$

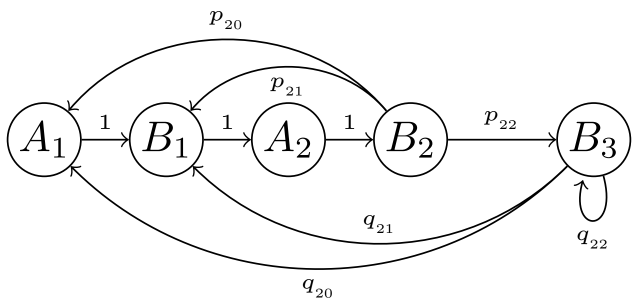


Figure 4.8: Schematic representation of the server states and transitions when $K = 2$.

(4.10)

if $n_1 + n_2 + m_1 + m_2 + m_3 = N$*, with* $(\gamma_1, \gamma_2, \kappa_1, \kappa_2, \kappa_3) = (p_{20} + \frac{p_{22}q_{20}}{1-q_{22}}, 1, 1, 1, \frac{p_{22}}{1-q_{22}})$ *and normalization constant*

$$H_N = \sum_{v_1+v_2+w_1+w_2+w_3=N} \frac{(v_1+v_2)!}{v_1!v_2!} \left(\frac{\gamma_1}{\lambda N}\right)^{v_1} \left(\frac{\gamma_2}{\lambda N}\right)^{v_2} \frac{(\kappa_1\tau_1)^{w_1}}{w_1!} \frac{(\kappa_2\tau_2)^{w_2}}{w_2!} \frac{(\kappa_3\tau_3)^{w_3}}{w_3!},$$

where n_i *is the number of open servers in state* A_i *and* m_i *the number of closed servers in state* B_i*.*

The proof of Proposition 4.3 is provided in Section 4.7.

The equilibrium distribution (4.10) can be simplified when only the number of open and closed servers are counted, as shown in the next corollary.

Corollary 4.3.

- *The equilibrium probability of there being* n *open servers and* $N - n$ *closed servers under the extension equals*

$$\pi(n, N-n) = \frac{\left(\frac{\gamma_1+\gamma_2}{\lambda N}\right)^n \frac{(\kappa_1\tau_1+\kappa_2\tau_2+\kappa_3\tau_3)^{N-n}}{(N-n)!}}{\sum_{w=0}^N \left(\frac{\gamma_1+\gamma_2}{\lambda N}\right)^w \frac{(\kappa_1\tau_1+\kappa_2\tau_2+\kappa_3\tau_3)^{N-w}}{(N-w)!}}.$$

  In particular, because of the PASTA property, the blocking probability is given by

$$\pi(0, N) = \frac{\frac{(xN)^N}{N!}}{\sum_{w=0}^N \frac{(xN)^w}{w!}},$$

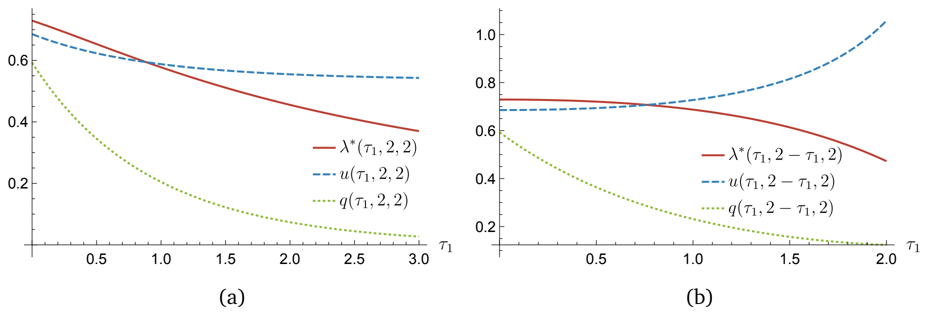


Figure 4.9: Maximum throughput λ^*, average number of updates per admitted job u and average queue position of admitted jobs q as a function of τ_1.

with $x = \lambda \times \frac{\kappa_1\tau_1+\kappa_2\tau_2+\kappa_3\tau_3}{\gamma_1+\gamma_2}$*, and* $\pi(0,N) \to \max\{0, 1-\lambda^*(\tau_1,\tau_2,\tau_3)/\lambda\}$ *as* $N \to \infty$*, which equals zero when* $\lambda \le \lambda^*(\tau_1,\tau_2,\tau_3) := \frac{\gamma_1+\gamma_2}{\kappa_1\tau_1+\kappa_2\tau_2+\kappa_3\tau_3}$.

- *The average number of messages per admitted job equals*

$$u(\tau_1,\tau_2,\tau_3) := \frac{\kappa_2+\kappa_3}{\gamma_1+\gamma_2}.$$

- *The average queue position of an admitted job equals*

$$q(\tau_1,\tau_2,\tau_3) := \frac{\mathrm{e}^{-\tau_1}}{\gamma_1+\gamma_2}.$$

The last two statements follow directly from the relative throughput values. These exact expressions for the maximum throughput λ^*, the average number of updates per admitted job u and the average queue position q of admitted jobs, allow us to evaluate the performance of this extension.

In Figure 4.9a, the value of τ_1 is varied while the values of τ_2 and τ_3 are kept constant. Since τ_1 represents the time that a server is closed when it has one job in queue, the result is that the second job that is sent to the server experiences a shorter queue in expectation. Indeed, for larger values of τ_1, the mean experienced queue length q decreases. As a further benefit, the mean number of updates decreases as well, since an idle server will take at least $\tau_1+\tau_2$ time units to be updated. The penalty incurred for these advantages is that the maximum throughput, λ^*, drops below the value of $\lambda^*(\delta,K)$ as asymptotically achieved by the baseline version of the hyper-scalable scheme, since servers may become idle during the τ_1 time in which they will not receive any more jobs.

Finally, in Figure 4.9b we show that a trade-off between the parameters is possible. τ_1 is increased while τ_2 is decreased, and this leads to interesting behavior. Around the point $\tau_1 = 0$, the values of λ^* and u do not change when the parameters are altered, but the value of q does change. Such a trade-off might be worth it in scenarios where mean queue lengths play an important role.

4.7 Closed queueing network and further proofs

In this section we establish product-form distributions for a more general closed queueing network scenario which captures the network representations of the hyper-scalable scheme and the extension considered in the previous section as special cases. This provides the proofs of Propositions 4.2 and 4.3.

The closed queueing network consists of N customers circulating among one single-server node and B infinite-server nodes. Customers can be of A classes while at the single-server node, denoted by $A_1, \ldots, A_A$. Denote the infinite-server nodes by $B_1, \ldots, B_B$. The routing probabilities are denoted by $p_{x\to y}$; this is the probability that a customer transitions from x to y (x and y may correspond to either a class or an infinite-server node).

Service completions at the multi-class single-server node occur at an exponential rate λN. The customer that completes service is either selected uniformly at random, or in a FCFS manner, where the next customer is the one that transitioned last. If the selected customer is of class i, then it immediately returns to the single-server node as a class-j customer with probability $p_{A_i\to A_j}$ or it moves to node B_j with probability $p_{A_i\to B_j}$. The service times at the infinite-server node B_i are deterministic and equal to τ_i. Upon completing service at node B_i, a customer moves to the single-server node as a class-j customer with probability $p_{B_i\to A_j}$, or to node B_j with probability $p_{B_i\to B_j}$.

The relative throughput values may be calculated from the traffic equations,

$$\begin{cases} \gamma_i = \sum_{j=1}^{A} p_{A_j\to A_i} \times \gamma_j + \sum_{j=1}^{B} p_{B_j\to A_i} \times \kappa_j, \\ \kappa_i = \sum_{j=1}^{A} p_{A_j\to B_i} \times \gamma_j + \sum_{j=1}^{B} p_{B_j\to B_i} \times \kappa_j, \end{cases}$$

where γ_i stands for the relative throughput of class A_i at the single-server node and κ_i for the relative throughput at node B_i. We assume a “single-chain network", where the routing probability matrix is irreducible, meaning that all customers can reach all classes and nodes.

Proposition 4.4.

(a) The equilibrium distribution of the system with N customers is

$$\pi(n_1, n_2, \ldots, n_A, m_1, m_2, \ldots, m_B) = F_N^{-1} \frac{(n_1 + \ldots + n_A)!}{n_1! \cdots n_A!} \prod_{i=1}^{A} \left(\frac{\gamma_i}{\lambda N}\right)^{n_i} \prod_{j=1}^{B} \frac{(\kappa_j \tau_j)^{m_j}}{m_j!} \tag{4.11}$$

if $n_1 + \ldots + n_A + m_1 + \ldots + m_B = N$, with normalization constant

$$F_N = \sum_{v_1 + \ldots + v_A + w_1 + \ldots + w_B = N} \frac{(v_1 + \ldots + v_A)!}{v_1! \cdots v_A!} \prod_{i=1}^{A} \left(\frac{\gamma_i}{\lambda N}\right)^{v_i} \prod_{j=1}^{B} \frac{(\kappa_j \tau_j)^{w_j}}{w_j!},$$

where n_i is the number of customers of class A_i at the single-server node and m_j the number of customers at infinite-server node B_j.

(b) The equilibrium probability of there being n customers at the single-server node and $N - n$ customers in total at all the infinite-server nodes equals

$$\begin{aligned} \pi(n, N-n) &= \sum_{\substack{n_1 + \ldots + n_A = n \\ m_1 + \ldots + m_B = N-n}} \pi(n_1, \ldots, n_A, m_1, \ldots, m_B) \\ &= \frac{\left(\frac{\sum_{i=1}^{A} \gamma_i}{\lambda N}\right)^{n} \frac{\left(\sum_{j=1}^{B} \kappa_j \tau_j\right)^{N-n}}{(N-n)!}}{\sum_{w=0}^{N} \left(\frac{\sum_{i=1}^{A} \gamma_i}{\lambda N}\right)^{w} \frac{\left(\sum_{j=1}^{B} \kappa_j \tau_j\right)^{N-w}}{(N-w)!}}. \end{aligned} \tag{4.12}$$

In particular, with $R = \frac{\gamma_1 + \ldots + \gamma_A}{\kappa_1 \tau_1 + \ldots + \kappa_B \tau_B}$ and $x = \lambda / R$, because of the PASTA property, the probability that no customer resides at the single-server node is

$$\pi(0, N) = \frac{\frac{(xN)^N}{N!}}{\sum_{w=0}^{N} \frac{(xN)^w}{w!}},$$

and $\pi(0, N) \to \max\{0, 1 - R/\lambda\}$ as $N \to \infty$ which equals zero when $\lambda \leq R$.

In order to prove Proposition 4.4, we will verify that the equilibrium distribution (4.11) satisfies the balance equations of the closed queueing network.

4.7.1 Proof of Proposition 4.4

In order to verify the balance equations, we may assume that the service times of the infinite-server nodes are exponentially distributed even though in our closed queueing network, the service times are deterministic. This is because the equilibrium distribution (4.11) is insensitive to the service time distribution of nodes and only depends on the means of them (see Chapter 3 of [Krz11] for a further discussion on this).

To see this, consider one infinite-server node D with exponential service rate μ_D and throughput value κ_D. This node adds the term

$$\frac{(\kappa_D/\mu_D)^d}{d!} \tag{4.13}$$

to the product-form equilibrium distribution, representing the presence of d customers in the infinite-server node. We now replace this infinite-server node by a series of infinite-server nodes, denoted by $D_1,\ldots,D_M$, each with an exponential service rate $M\mu_D$. The transition probabilities are altered in such a way that every transition previously to node D, now transitions to node D_1 instead. Customers then transition from node D_i to D_{i+1} for $i = 1,\ldots,D-1$ with probability one. Finally, any transition previously from node D, will now transition from node D_M. This construction makes every customer stay in this collection of nodes for M exponentially distributed phases, which is an Erlang$(M, M\mu)$ distributed random variable. All other throughput values in the network remain equal.

The throughput values of all these nodes will be equal to κ_D (since they are in series). Finally, similarly to the simplification of (4.11) to (4.12), all nodes $D_1,\ldots,D_M$ may be aggregated, which would lead to a term

$$\frac{(\sum_{i=1}^{M}\kappa_D/(M\mu_D))^d}{d!} = \frac{(\kappa_D/\mu_D)^d}{d!}$$

in the equilibrium probability, representing the presence of d customers in total in the infinite-server nodes $D_1,\ldots,D_M$. Note that the term in the RHS exactly matches the term (4.13), that appears when the node D has an exponentially distributed service time. This shows that the equilibrium distribution does not change when an exponential node is replaced by an Erlang$(M, M\mu)$ node, for any integer M, which can also be verified by substitution in the balance equations. Of course, each infinite-server node B_i with $\mu_i = 1/\tau_i$ can be replaced by such an Erlang distribution using this construction.

Because an Erlang$(M, M\mu)$ random variable converges to a deterministic quantity $1/\mu$ as M tends to infinity, this indicates that the equilibrium distribution also holds with infinite-server nodes that have deterministic service times. In fact, the node D may be replaced by any phase-type distribution, and every distribution may be approximated arbitrarily closely by phase-type distributions, implying that the equilibrium distribution in (4.11) in fact holds for generally distributed service times with mean τ_i at the infinite-server node B_i as well, although that is not directly relevant for our purposes.

We will now verify that (4.11) indeed solves the balance equations for the random order of service case, and we will use $\mu_i = 1/\tau_i$, representing the rates of the infinite-server nodes. The proof for the FCFS case is quite similar, but involves a more detailed state representation.

Proof of part (a) - ROS. We denote by (a, b) the vector $(a_1, \ldots, a_A, b_1, \ldots, b_B)$ and by e_i the i-th unit vector.

Note that (4.11) is a proper distribution by definition. Since the equilibrium distribution is unique, it suffices to verify that (4.11) satisfies the following set of balance equations:

$$\begin{aligned}
&\left(\mathbb{1}\{a_1 + \ldots + a_A > 0\}\lambda N + b_1\mu_1 + \ldots + b_B\mu_B\right)\pi(a, b) \\
&= \sum_{i=1}^{A}\sum_{j=1}^{A} \mathbb{1}\{a_j > 0\} p_{A_i \to A_j} \frac{a_i + \mathbb{1}\{i \neq j\}}{a_1 + \ldots + a_A} \lambda N \pi(a + e_i - e_j, b) \\
&+ \sum_{i=1}^{A}\sum_{j=1}^{B} \mathbb{1}\{b_j > 0\} p_{A_i \to B_j} \frac{a_i + 1}{a_1 + \ldots + a_A + 1} \lambda N \pi(a + e_i, b - e_j) \\
&+ \sum_{i=1}^{B}\sum_{j=1}^{A} \mathbb{1}\{a_j > 0\} p_{B_i \to A_j} (b_i + 1)\mu_i \pi(a - e_j, b + e_i) \\
&+ \sum_{i=1}^{B}\sum_{j=1}^{B} \mathbb{1}\{b_j > 0\} p_{B_i \to B_j} (b_i + \mathbb{1}\{i \neq j\})\mu_i \pi(a, b + e_i - e_j).
\end{aligned}$$

The first line of the RHS refers to transitions where a customer at the single-server node transitions to the same node and may change class. The second line refers to transitions from the single-server node to one of the infinite-server nodes. Lines three and four correspond to transitions from an infinite-server node, to the single-server node or to another infinite-server node, respectively.

We will show that (4.11) satisfies the balance equations. By using the definition of (4.11) in the RHS, we obtain

$$\sum_{i=1}^{A}\sum_{j=1}^{A}\mathbb{1}\{a_j>0\}p_{A_i\to A_j}\frac{a_i+1}{a_1+\ldots+a_A}\lambda N\pi(a,b)\frac{a_j}{a_i+1}\frac{\gamma_i}{\lambda N}\frac{\lambda N}{\gamma_j}$$
$$+\sum_{i=1}^{A}\sum_{j=1}^{B}\mathbb{1}\{b_j>0\}p_{A_i\to B_j}\frac{a_i+1}{a_1+\ldots+a_A+1}\lambda N\pi(a,b)\frac{a_1+\ldots+a_A+1}{a_i+1}\frac{\gamma_i}{\lambda N}\frac{b_j}{\kappa_j T_j}$$
$$+\sum_{i=1}^{B}\sum_{j=1}^{A}\mathbb{1}\{a_j>0\}p_{B_i\to A_j}(b_i+1)\mu_i\pi(a,b)\frac{a_j}{a_1+\ldots+a_A}\frac{\lambda N}{\gamma_j}\frac{\kappa_i\tau_i}{b_i+1}$$
$$+\sum_{i=1}^{B}\sum_{j=1}^{B}\mathbb{1}\{b_j>0\}p_{B_i\to B_j}(b_i+1)\mu_i\pi(a,b)\frac{\kappa_i/\mu_i}{b_i+1}\frac{b_j}{\kappa_j/\mu_j}.$$

Next, we combine the inside sums, resulting in

$$\sum_{j=1}^{A}\mathbb{1}\{a_j>0\}\frac{a_j}{a_1+\ldots+a_A}\frac{\lambda N}{\gamma_j}\left[\sum_{i=1}^{A}p_{A_1\to A_j}\gamma_i+\sum_{i=1}^{B}p_{B_i\to A_j}\kappa_i\right]\pi(a,b)$$
$$+\sum_{j=1}^{B}\mathbb{1}\{b_j>0\}\frac{b_j}{\kappa_j/\mu_j}\left[\sum_{i=1}^{A}p_{A_i\to B_j}\gamma_i+\sum_{i=1}^{B}p_{B_i\to B_j}\kappa_i\right]\pi(a,b)$$
$$=\left[\sum_{j=1}^{A}\mathbb{1}\{a_j>0\}\frac{a_j}{a_1+\ldots+a_A}\lambda N+\sum_{j=1}^{B}b_j\mu_j\right]\pi(a,b)$$
$$=\left(\mathbb{1}\{a_1+\ldots+a_A>0\}\lambda N+\sum_{j=1}^{B}b_j\mu_j\right)\pi(a,b).$$

□

Proof of part (a) - FCFS. The proof for the FCFS case consists of multiple steps. First we define a more detailed state space.

Extended state space. A state is represented by $(c,b)=((c_1,\ldots,c_m),(b_1,\ldots,b_B))$, which represents the situation where m customers are at the single-server node, and the order of the classes of customers is saved as well: the kth customer at the single-server node has class c_k. We will sometimes refer to the number of customers of a specific class with $a_i=\sum_j\mathbb{1}\{c_j=A_i\}$. Furthermore, b_i customers are at the infinite-server node B_i.

Equilibrium distribution for the extended state space. We will show that the equilibrium distribution (modulo normalization constant) of state (c,b) equals

$$\tilde{\pi}(c,b)=\left(\frac{\gamma_1}{\lambda N}\right)^{a_1}\cdots\left(\frac{\gamma_A}{\lambda N}\right)^{a_A}\frac{(\kappa_1/\mu_1)^{b_1}}{b_1!}\cdots\frac{(\kappa_B/\mu_B)^{b_B}}{b_B!} \qquad (4.14)$$

with a_k the number of customers of class A_k.

We assume FCFS arrivals of customers at the single-server node: customers arrive at the end of the line at the single-server node and only the customer first in line is able to transition.

Balance equations. First, we introduce the balance equations, in which the symbol m is used to denote the length of vector c,

$$\begin{aligned}
&\big(\mathbb{1}\{m>0\}\lambda N + b_1\mu_1 + \ldots + b_B\mu_B\big)\tilde{\pi}(c,b) \\
&= \sum_{i=1}^{A} \mathbb{1}\{m>0\} p_{A_i\to c_m} \lambda N \tilde{\pi}((A_i, c_1,\ldots,c_{m-1}), b) \\
&+ \sum_{i=1}^{A}\sum_{j=1}^{B} \mathbb{1}\{b_j>0\} p_{A_i\to B_j} \lambda N \tilde{\pi}((A_i, c_1,\ldots,c_m), b-e_j) \\
&+ \sum_{i=1}^{B} \mathbb{1}\{m>0\} p_{B_i\to c_m} (b_i+1)\mu_i \tilde{\pi}((c_1,\ldots,c_{m-1}), b+e_i) \\
&+ \sum_{i=1}^{B}\sum_{j=1}^{B} \mathbb{1}\{b_j>0\} p_{B_i\to B_j} (b_i + \mathbb{1}\{i\neq j\})\mu_i \tilde{\pi}(c, b+e_i-e_j).
\end{aligned} \tag{4.15}$$

The term before $\pi(c,b)$ on the LHS represents the outgoing rate of state (c,b), which equals a rate of λN for the single-server node (if at least one customer is present there) plus a rate of $b_j\mu_j$, for each infinite-server node B_j.

On the RHS, four possible transitions to state (c,b) are shown preceded by the rate of the transitions. First, a transition from the non-empty single-server node makes the then first customer change its class from c_{m-1} to class c_m. If the previous class order at the single-server node is $c_m-1, c_1,\ldots,c_{m-1}$, then a transition to that node will make the class order exactly c. Second, if the previous class order at the single-server node is $K-1, c_1,\ldots,c_m$, then a transition from that node to an infinite-server node will make the class order exactly c. Additionally, if the number of customers at infinite-server node B_j was b_j-1, then it will become b_j as the infinite-server node receives an extra customer. Third, any of the customers at the infinite-server nodes might transition to the single-server node. Finally, customers might transition from and to one of the infinite-server nodes.

We will show that $\tilde{\pi}$ satisfies the balance equations. By using the definition of $\tilde{\pi}$ in the RHS, we obtain

$$\begin{aligned}
&\sum_{i=1}^{A} \mathbb{1}\{m>0\} p_{A_i\to c_m} \lambda N \tilde{\pi}(c,b) \frac{\gamma_i}{\lambda N} \frac{\lambda N}{\gamma_{c_m}} \\
&+ \sum_{i=1}^{A} \sum_{j=1}^{B} \mathbb{1}\{b_j>0\} p_{A_i\to B_j} \lambda N \tilde{\pi}(c,b) \frac{\gamma_i}{\lambda N} \frac{b_j}{\kappa_j \tau_j} \\
&+ \sum_{i=1}^{B} \mathbb{1}\{m>0\} p_{B_i\to c_m} (b_i+1)\mu_i \tilde{\pi}(c,b) \frac{\lambda N}{\gamma_{c_m}} \frac{\kappa_i/\mu_i}{b_i+1} \\
&+ \sum_{i=1}^{B} \sum_{j=1}^{B} \mathbb{1}\{b_j>0\} p_{B_i\to B_j} (b_i+1)\mu_i \tilde{\pi}(c,b) \frac{\kappa_i/\mu_i}{b_i+1} \frac{b_j}{\kappa_j/\mu_j}.
\end{aligned} \tag{4.16}$$

Next, we reorganize terms, yielding

$$\begin{aligned}
&\mathbb{1}\{m>0\} \frac{\lambda N}{\gamma_{c_m}} \tilde{\pi}(c,b) \left[\sum_{i=1}^{A} p_{A_i\to c_m} \gamma_i + \sum_{i=1}^{B} p_{B_i\to c_m} \kappa_i \right] \\
&+ \sum_{j=1}^{B} \mathbb{1}\{b_j>0\} \frac{b_j}{\kappa_j/\mu_j} \tilde{\pi}(c,b) \left[\sum_{i=1}^{A} p_{A_i\to B_j} \gamma_i + \sum_{i=1}^{B} p_{B_i\to B_j} \kappa_i \right] \\
&= \left[\mathbb{1}\{m>0\} \lambda N + \sum_{j=1}^{B} b_j \mu_j \right] \tilde{\pi}(c,b),
\end{aligned} \tag{4.17}$$

which shows that $\tilde{\pi}$ is the equilibrium distribution of the extended state space.

Finally, note that in the original state space, only the number of customers of certain classes is tracked. Thus, $\pi(a,b)$ is an enumeration of $\tilde{\pi}(c,b)$ over all possible orders with the correct number of customers of a certain class. The number of possible orders is $\binom{a_1+\ldots+a_A}{a_1\ldots a_A}$, which leads to $\pi(a,b) = \binom{a_1+\ldots+a_A}{a_1\ldots a_A}\pi(c,b)$; the description of π as presented in the statement of the proposition. □

4.8 Conclusion

We established a universal upper bound for the achievable throughput of any dispatcher-driven algorithm for a given communication budget and queue limit. We also introduced a specific hyper-scalable scheme which can operate at any given message rate and enforce any given queue limit, while allowing the system dynamics to be captured via a closed product-form network. We leveraged the product-form distribution to show that the bound is tight, and that the proposed hyper-scalable scheme provides asymptotic optimality in the three-way trade-off among performance, communication and throughput. Extensive simulation experiments were presented to illustrate the results and make comparisons with various alternative design options.

The work-conserving variant covered in Section 4.5.3 is especially worth discussing further. Intuitively, letting servers work all the time seems better than pausing the servers when they become open, but this remains to be rigorously proven. It also seems straightforward to prove this, possibly by coupling the two systems in a specific way. However, we were not able to do so, and moreover, we discovered some sample paths in which it is seemingly better to stop the servers from working once in a while, to avoid placing jobs in unfortunate positions. To prove that the work-conserving discipline outperforms the standard one, it seems that a one-to-one coupling is not possible, and one would need to look at the average behavior of both systems, which seems tedious. Note that this variant only outperforms the baseline scenario in finite systems, as the hyper-scalable scheme remains asymptotically optimal.

The extension aimed at minimizing waiting times that was introduced in Section 4.6 warrants further attention as well. For the baseline scenario, we were able to prove a strict relationship between the amount of communication and the throughput. Likewise, there might exist a result, similar in spirit to Theorem 4.1, which provides an upper bound for the throughput *and* the average queue position of admitted jobs, given a certain communication budget. The main point of concern in this regard is that the concavity argument no longer seems to hold.

Finally, it could be worth investigating whether the current framework could be broadened further. It may be possible for example to extend the category of algorithms considered, specifically allowing for pull-based schemes. While the results in Chapter 5 will imply that Theorem 4.1 does not hold for pull-based schemes, there might be a larger upper bound covering such algorithms as well. For further extensions, other performance metrics might be considered too, such as the mean waiting time as opposed to the throughput subject to a queue limit.

Chapter 5

Load balancing with negative acknowledgments

Based on:

[BZB20] M. van der Boor, M. Zubeldia, and S. C. Borst. "Zero-wait load balancing with sparse messaging". In: *Operations Research Letters* 48.3 (2020), pp. 368–375

5.1 Introduction

In this chapter we introduce a hyper-scalable server-driven scheme, called Join-the-Open-Queue (JOQ), which allows for vanishing queueing delays, while using minimal communication overhead. The JOQ scheme exploits the crucial insight that an idle state need not be explicitly signaled by using a message, but can also be implicitly inferred by the dispatcher when *not receiving a message* from a server at a pre-arranged time instant. The basic principle is somewhat similar in spirit to negative acknowledgments in end-to-end transport protocols, but to the best of our knowledge has not been adopted in a load balancing context so far. This paradigm allows for an even lower communication overhead than the schemes in Chapters 3 and 4. The differences between schemes from these chapters and the JOQ scheme may seem quite large at first sight, but there are in fact striking similarities, which we will discuss after we explain the JOQ scheme in greater detail in Section 5.3.

In the JOQ scheme, servers send busy alerts to the dispatcher at pre-arranged time instants as long as they have uncompleted jobs. The status of a server is *closed* or *open*, depending on whether or not the dispatcher received a busy alert from the server at the most recent pre-arranged time instant, respectively, implying that an *open* server is guaranteed to be *idle*. The dispatcher assigns incoming jobs to an open server whenever there are any, or to a uniformly at random selected server otherwise.

For convenience, we focus on the case where the pre-arranged time instants are set at fixed intervals of length τ after an open server receives a job from the dispatcher. In that case, the system dynamics under the JOQ scheme evolve in a similar manner as under the JIQ scheme with service requirements inflated to be multiples of τ. In particular, the waiting probability under the JOQ scheme is upper bounded by that under the JIQ scheme with the inflated service requirements. Moreover, for any load below a certain threshold $\lambda^* < 1$, vanishing queueing delay is also achieved by the JOQ scheme for a suitable choice of τ with strictly less than one message per job. It turns out that there is a range of values for τ, depending on the arrival rate, for which the message rate, probability of queueing and the mean delays are small simultaneously. The number of messages per job in fact even tends to zero when the load approaches zero. Moreover, we show how a combination of the JIQ and JOQ schemes provides a way to achieve a vanishing queueing delay with strictly less than one message per job for any subcritical load.

The remainder of this chapter is organized as follows. In Section 5.2 we present a detailed model description and algorithm specification, and we summarize the main contribution of this chapter. In Section 5.3 we analyze the JOQ scheme and state for what parameter values it achieves a vanishing queueing delay and uses less than one message per job on average. In Section 5.4 we introduce the system design where jobs are distributed over servers executing JOQ and servers using JIQ. In Section 5.5 we provide numerical simulations (see Section 1.3.3) and discuss various broader issues and observations. Section 5.6 concludes and mentions several future research avenues.

5.2 Model description and key results

We consider a system that consists of N identical servers, each with an infinite-buffer FIFO queue. Jobs arrive according to a Poisson process with rate λN ($\lambda < 1$) to a central dispatcher, and need to be forwarded to one of the N servers immediately upon arrival. The job sizes are independent and generally distributed with unit mean. Moreover, we assume that the job sizes are not known to the dispatcher and servers, and that both dispatcher and servers can make use of unlimited memory to store information.

We now introduce the **Join-the-Open-Queue (JOQ)** algorithm, and discuss its behavior and implementation. In the JOQ algorithm, servers periodically communicate their status to the dispatcher at predetermined times, also referred to as update epochs. Normally, servers would send a message to the dispatcher to advertise their idleness, but in the JOQ algorithm, servers send messages to the dispatcher when they are *not* idle. If the dispatcher receives no message from a server at one of the server's update epochs, then its idleness implicitly becomes known to the dispatcher, without requiring any explicit messaging. In particular, a server is said to be *open* when the dispatcher knows that it is idle, and *closed* otherwise.

We now provide a more detailed description of the JOQ algorithm, specifying how the dispatcher communicates with the servers, and how dispatching decisions are made.

Communication with update epochs: *When a server is open and receives a job, it becomes closed and an "update epoch" for the server is scheduled at τ time units in the future. Moreover, each server maintains a virtual queue with the same arrivals as its real queue, but where a job of size x in its real queue has size $\tau\lceil x/\tau\rceil \geq x$ in its virtual one. At the next update epoch of a server, if its virtual queue is not empty, then it sends a message to the dispatcher to advertise its closedness, and a new update epoch for the server is scheduled at τ time units in the future. Otherwise, if its virtual queue is empty, then no message is sent and the server becomes open.*

Dispatching rule: *If a job arrives and there are open servers, then the job is sent to an open server chosen uniformly at random. Otherwise, the job is sent to any of the servers chosen uniformly at random.*

Note that the JOQ algorithm is emulating the JIQ algorithm but using open and closed servers instead of idle and busy servers. However, while open servers are surely idle, closed servers are not necessarily busy, so there might be idle servers when there are no open servers. This discrepancy stems from the fact that there is a delay between the time a server becomes idle, and the moment the dispatcher becomes aware that the server is idle, due to the use of predetermined update epochs.

A further discussion on design choices for the JOQ algorithm can be found in Section 5.5, where we use simulation results to put some of the choices into perspective.

The main results may be summarized as follows.

1. For the JOQ algorithm, when the arrival rate is below a certain threshold, the steady-state queueing delay of a typical job vanishes in the many-server limit.

2. For the JOQ algorithm, when the arrival rate is below a certain threshold, the expected number of messages per job is less than one for all N large enough, and tends to zero when the arrival rate approaches zero.

3. For a mixture of the JOQ and JIQ algorithms, when $\lambda < 1/2$ (or $\lambda < 1$ and the job sizes have decreasing hazard rate), the steady-state queueing delay of a typical job vanishes in the many-server limit and the expected number of messages per job is less than one for all N large enough.

5.3 Performance of the JOQ algorithm

In this section we present our main results regarding the stability, delay performance and message rate of the JOQ algorithm introduced in the previous section.

First, we formalize some important concepts. We say that a dispatching algorithm has *vanishing queueing delay* if the queue state process is stable for all N large enough, and if the probability of having positive queueing delay and the expected queueing delay both converge to zero, as $N \to \infty$. Moreover, as a measure of the communication overhead, we use the expected number of messages per job. While for algorithms such as JSQ(d) and JIQ it is straightforward to tie messages to jobs, for our JOQ algorithm this is not as immediate. We consider a message to be tied to a job if it is sent while the corresponding virtual job is in service in the virtual queue. Then, we define the *expected number of messages per job* as the steady-state expectation of this random integer, for a typical job.

The following theorem states for what values of τ the JOQ algorithm has vanishing queueing delay, and provides an expression for the limit of the expected number of messages per job.

Theorem 5.1. *For every τ such that $\lambda\mathbb{E}[\tau\lceil X/\tau\rceil] < 1/2$, the following properties hold for the JOQ algorithm with that value of τ:*

i) it has vanishing queueing delay,

ii) the expected number of messages per job converges to $\mathbb{E}[\lfloor X/\tau \rfloor]$, as $N \to \infty$.

When $\lambda < 1/2$, a value of $\tau > 0$ exists such that $\lambda\mathbb{E}[\tau\lceil X/\tau\rceil] < 1/2$ regardless of the job size distribution, since $\lambda\mathbb{E}[\tau\lceil X/\tau\rceil] \to \mathbb{E}[X]$ as $\tau \downarrow 0$.

Proof of Theorem 5.1.

i) First note that the real queues are stochastically dominated by the virtual queues, as arrivals happen simultaneously to each real and corresponding virtual queue, and jobs require strictly more service in the virtual queues. Thus, it is enough to establish vanishing queueing delay in the virtual queues.

We now focus on the virtual queues. Since all jobs in the virtual queues have sizes multiple of τ, a virtual queue can only become empty at an update epoch. When the virtual queue becomes empty, no message is sent, implicitly letting the dispatcher know immediately that it is empty, as it would have received a message otherwise. As a result, the dispatcher is aware at all times which virtual queues are empty (i.e., which servers are open) and which ones are not. Combining this with the fact that the dispatching decisions of the JOQ algorithm mimic the ones of the JIQ algorithm but using open/closed status of servers instead of idle/busy, we see that the virtual queues behave exactly as the queues of a system running the JIQ algorithm with job sizes distributed as $\tau\lceil X/\tau\rceil$. Since the expected size of a job in the virtual queues is $\mathbb{E}[\tau\lceil X/\tau\rceil]$, their load is $\lambda\mathbb{E}[\tau\lceil X/\tau\rceil] < 1/2$. Thus, Theorem 2 in [FS17] and the PASTA property imply that the queue state process of the virtual queues is stable for all N large enough, and that the probability of a typical virtual job having positive queueing delay vanishes as $N \to \infty$. Combining this with Lemma 4 in [FS17], we conclude that the expected queueing delay of a typical virtual job vanishes as well.

ii) For the expected number of messages per job, we distinguish between jobs sent to open servers and jobs sent to closed servers. If a job of size x arrives at time 0 to an open server (which is always an idle server), then this job will be in service during the interval of time $[0, \tau\lceil x/\tau\rceil)$. During this interval of time, messages are sent every τ time units (excluding 0), which results in $\lfloor x/\tau \rfloor$ messages. On the other hand, suppose that a job of size x is assigned to a closed server (i.e., a server with a non-empty virtual queue). Since the server was closed when the job was assigned to it, the server will send a message to the dispatcher at the time that the virtual job started its service in the virtual queue.

After this, exactly one message is sent every τ time units while the virtual server is busy, similarly to when jobs are sent to open servers. In total, this results in $1 + \lfloor x/\tau \rfloor$ messages.

Denote by $p_o(N)$ the steady-state probability that there is at least one open server in the N-server system (which is known to exist by Theorem 2 in [FS17]). Moreover, note that the PASTA property implies that $p_o(N)$ is also the probability that a typical virtual job is sent to an open server. It follows that the expected number of messages per job is $p_o(N)\mathbb{E}[\lfloor X/\tau \rfloor] + [1 - p_o(N)](1 + \mathbb{E}[\lfloor X/\tau \rfloor])$. Since Theorem 2 in [FS17] also implies that $p_o(N) \to 1$ as $N \to \infty$, the expected number of messages per job converges to $\mathbb{E}[\lfloor X/\tau \rfloor]$, as $N \to \infty$. □

In light of Theorem 5.1, we would require $\lambda\mathbb{E}[\tau\lceil X/\tau \rceil] < 1/2$ and $\mathbb{E}[\lfloor X/\tau \rfloor] < 1$ for vanishing queueing delay while using less than one message per job in expectation. This notion is used in the next corollary, which shows for what values of τ this is accomplished when $\lambda < 1/4$.

Corollary 5.1. *Suppose that $\lambda < 1/4$. For all τ such that $1 < \tau < 1/(2\lambda) - 1$, the JOQ algorithm has vanishing queueing delay, and the expected number of messages per job is less than one, as $N \to \infty$.*

Note that such a value of τ always exists, since $1 < 1/(2\lambda) - 1$ when $\lambda < 1/4$.

Proof. Corollary 5.1 follows from Theorem 5.1 since $\lambda\mathbb{E}[\tau\lceil X/\tau \rceil] \le \lambda(\tau + \mathbb{E}[X]) = \lambda(\tau + 1) < \lambda(1/(2\lambda) - 1 + 1) = 1/2$ and $\mathbb{E}[\lfloor X/\tau \rfloor] \le \mathbb{E}[X]/\tau = 1/\tau < 1$. □

The JOQ algorithm has additional desirable properties, for example that the expected number of messages per job in the many-server limit can be driven to zero as $\lambda \downarrow 0$. To accomplish this, the value of $\tau = \tau(\lambda)$ needs to be adjusted appropriately given the load λ.

Corollary 5.2. *When the value of τ in the JOQ algorithm is chosen depending on λ such that $\tau(\lambda) < 1/(2\lambda) - 1$ and $\tau(\lambda) \to \infty$ as $\lambda \downarrow 0$, the JOQ algorithm has vanishing queueing delay, and the expected number of messages per job in the many-server limit tends to zero as $\lambda \downarrow 0$.*

Note that the convergence of the expected number of messages per job to zero refers to first taking the limit as $N \to \infty$, and then the limit as $\lambda \downarrow 0$.

Proof. Note that Corollary 5.1 guarantees a vanishing queueing delay since $\tau(\lambda) < 1/(2\lambda) - 1$ for any λ, so also more specifically for the limit when $\lambda \downarrow 0$. Moreover, Theorem 5.1 states that the expected number of messages per job in the many-server limit converges to $\mathbb{E}[\lfloor X/\tau \rfloor]$, with $\mathbb{E}[\lfloor X/\tau(\lambda) \rfloor] \le \mathbb{E}[X]/\tau(\lambda) = 1/\tau(\lambda) \downarrow 0$ as $\lambda \downarrow 0$. □

Another desirable property of the JOQ algorithm is that, although Corollary 5.1 requires $\lambda < 1/4$ in order to guarantee a vanishing queueing delay with less than one message per job in expectation, this can be achieved for higher values of λ for some job size distributions. In particular, for job size distributions with decreasing hazard rate, we have the following.

Corollary 5.3. *Suppose that the job sizes X have a decreasing hazard rate and that $\mathbb{P}(X > 1/(4\lambda)) < 1/2$. Then, the JOQ algorithm with $\tau = 1/(4\lambda)$ has vanishing queueing delay, and the expected number of messages per job is less than one in the many-server limit.*

Proof. When X has a well-defined and decreasing hazard rate, its density function $f(t)$ is decreasing since $f(t+u) < f(t)\frac{1-F(t+u)}{1-F(t)} < f(t)$. This implies that the distribution function $F(t)$ is concave and $1-F(t)$ is convex, which gives $\frac{1-F((k+1)\tau)}{1-F(k\tau)} \leq \frac{1-F(k\tau)}{1-F((k-1)\tau)}$. Successively applying this inequality gives $\frac{1-F((k+1)\tau)}{1-F(k\tau)} \leq 1-F(\tau)$ for any integer k. This leads to the inequality

$$\mathbb{P}(X > k\tau) = 1 - F(k\tau) = \prod_{i=0}^{k-1} \frac{1-F((i+1)\tau)}{1-F(i\tau)} \leq (1-F(\tau))^k = (\mathbb{P}(X > \tau))^k. \tag{5.1}$$

Using this inequality, it follows that

$$\mathbb{E}[\tau\lceil X/\tau\rceil] = \tau\sum_{k=0}^{\infty} \mathbb{P}(X > k\tau) \leq \tau\sum_{k=0}^{\infty} (\mathbb{P}(X > \tau))^k = \frac{\tau}{1-\mathbb{P}(X > \tau)} < \frac{1}{2\lambda}, \tag{5.2}$$

where in the last inequality we used that $\tau = 1/(4\lambda)$ and that $\mathbb{P}(X > 1/(4\lambda)) < 1/2$. This allows us to use Theorem 5.1 showing that the queueing delay vanishes. Furthermore, the expected number of messages per job is

$$\mathbb{E}[\lfloor X/\tau\rfloor] = \sum_{k=1}^{\infty} \mathbb{P}(X > k\tau) \leq \sum_{k=1}^{\infty} (\mathbb{P}(X > \tau))^k = \frac{\mathbb{P}(X > \tau)}{1-\mathbb{P}(X > \tau)} < 1, \tag{5.3}$$

where in the last inequality we once again used that $\tau = 1/(4\lambda)$ and that $\mathbb{P}(X > 1/(4\lambda)) < 1/2$. This concludes the proof. □

For the special case of exponential job sizes, a vanishing queueing delay and less than one message per job in expectation can be obtained when $\lambda < 1/\ln(16) \approx 0.36$. This is shown in the following result.

Corollary 5.4. *Suppose that the job size X is exponentially distributed and that $\lambda < 1/\ln(16)$. Then, there exists some τ such that the JOQ algorithm with that value of τ has vanishing queueing delay, and the expected number of messages per job is less than one in the many-server limit.*

Proof. First note that, for exponential job sizes, (5.1) holds with equality. Thus, Theorem 5.1 implies that a vanishing queueing delay and less than one message per job in expectation in the many-server limit is obtained as long as $\mathbb{E}[\tau\lceil X/\tau\rceil] = \frac{\tau}{1-\mathbb{P}(X>\tau)} = \frac{\tau}{1-e^{-\tau}} < \frac{1}{2\lambda}$ and $\mathbb{E}[\lfloor X/\tau\rfloor] = \frac{\mathbb{P}(X>\tau)}{1-\mathbb{P}(X>\tau)} = \frac{e^{-\tau}}{1-e^{-\tau}} < 1$. It can be checked that there exists a $\tau > 0$ such that these two inequalities are satisfied as long as $\lambda < 1/\ln(16)$. □

Remark. When the job size distribution is exponential, it is optimal to take the update epochs equidistant, see Section 5.A.

Remark. In [FS17] the authors show that the JIQ algorithm with general job size distribution has vanishing queueing delay as long as $\lambda < 1/2$. If it also has vanishing queueing delay whenever $\lambda < 1$, which is conjectured, then Theorem 5.1 would hold for all $\lambda < 1$ and for all τ such that $\lambda\mathbb{E}[\tau\lceil X/\tau\rceil] < 1$. This decreases the communication overhead since τ can be chosen larger. Moreover, such a stronger result would also improve the conditions stated in Corollaries 5.1 and 5.3. For example, Corollary 5.1 would hold for all $\lambda < 1/2$ and for all τ such that $1 < \tau < 1/\lambda - 1$.

We will briefly compare the JOQ algorithm with the hyper-scalable scheme from Chapter 4. Of course, one of the most significant differences is the use of implicit messaging in the JOQ algorithm. Nonetheless, the hyper-scalable algorithm for $K = 1$ behaves the same as the JOQ algorithm for exponentially distributed job sizes, except for when a job arrives to a busy server in the JOQ algorithm, as both algorithms will wait τ time units before 'updating' the server. This equivalence can also be seen in the analyses of the algorithms. The throughput of the hyper-scalable algorithm $\lambda^*(1/\tau, K)$ in (4.1) for $K = 1$ and the service rate of the system with the JOQ algorithm, $1/\mathbb{E}[\tau\lceil X/\tau\rceil]$, both equal $(1-e^{-\tau})/\tau$.

5.4 Hybrid system using the JOQ and JIQ algorithms

In this section we introduce a system design that uses a mixture of the JOQ and JIQ algorithms in order to show that a vanishing queueing delay and less than one message per job in expectation can be achieved for any load $< 1/2$ (or for any load < 1 when the job sizes have a decreasing hazard rate).

5.4.1 System design

In particular, we propose the following system design.

Hybrid JOQ/JIQ: *The N-server system is divided into two subsystems, one with a fixed fraction f of the servers running the JOQ algorithm with some value of τ (JOQ-subsystem), and one with a fixed fraction $1-f$ of the servers running the JIQ algorithm (JIQ-subsystem) (numbers may be rounded to obtain integers). When a job arrives, it is routed to the JOQ-subsystem with probability p, and to the JIQ-subsystem with probability $1-p$.*

Denote by ρ the maximum load for which the JIQ-subsystem is guaranteed to achieve a vanishing queueing delay, which is either 1 if the job size distribution has decreasing hazard rate (case 1; [Sto15]), or 1/2 otherwise (case 2; [FS17]).

Theorem 5.2. *Suppose that either X has decreasing hazard rate and $\lambda < 1$ (case 1), or that $\lambda < 1/2$ (case 2). Then the hybrid JOQ/JIQ system design has a vanishing queueing delay, and its expected number of messages per job is less than one in the many-server limit, when f, p, and τ are chosen such that $p > 0$, (a) $\lambda(1-p)/(1-f) < \rho$, (b) $\lambda p/f < 1/4$ and (c) $1 < \tau < f/(2\lambda p) - 1$. Such values for f, p and τ always exist.*

Proof of Theorem 5.2. Note that both subsystems can be analyzed separately, as the Poisson arrivals are split randomly. First, since the relative arrival rate to the JIQ-subsystem equals $\lambda(1-p)/(1-f)$, which is assumed to be strictly less than ρ by Equation (a), the JIQ-subsystem has a vanishing queueing delay. Furthermore, the JIQ-subsystem uses at most one message per job for all N by construction.

We now turn to the JOQ-subsystem. Since the relative arrival rate to this subsystem equals $\bar{\lambda} = \lambda p/f$, which is assumed to be less than 1/4 by Equation (b), and since Equation (c) is assumed to hold, Corollary 5.1 (applied with $\bar{\lambda}$) implies that the JOQ-subsystem has a vanishing queueing delay, and that it uses strictly less than one message per job in expectation in the many-server limit. Combining this with the assumption that $p > 0$, and the fact that the JIQ-subsystem also achieves a vanishing queueing delay with at most one message per job, we conclude that the whole system achieves a vanishing queueing delay, and that it uses strictly less than one message per job in expectation, in the many-server limit.

Finally, it remains to be checked that there exist parameters f, p, and τ that fulfill the requirements. Indeed, if we choose $f = 1 - \lambda$ for case 1 and $f = 1 - 2\lambda$ for case 2, thus fulfilling (a), then for every $\lambda < 1$ or for every $\lambda < 1/2$, we can pick $p > 0$ small enough so that (b) is satisfied. Moreover, since the upper bound for τ in (c) is larger than 1 for all p small enough and becomes arbitrarily large for small $p > 0$, condition (c) can also be satisfied for all $\lambda < 1$ or $\lambda < 1/2$. □

5.4.2 Choosing the system parameters

Since the parameters f, p, and τ to implement the Hybrid JOQ/JIQ algorithm are not uniquely determined by the parameters of the system, a natural choice for them is one that minimizes the expected number of messages per job. Although this quantity is unknown for any finite N, we can use its limiting values as $N \to \infty$ as a proxy. In particular, the parameters that minimize this limit are given by the solution to the following optimization problem.

$$\begin{aligned} \inf_{p,f,\tau} \quad & 1 - p + p\mathbb{E}[\lfloor X/\tau \rfloor] \\ \text{s.t.} \quad & \lambda p \tau \mathbb{E}[\lceil X/\tau \rceil] < f/2, \quad \lambda(1-p) < \rho(1-f), \\ & 0 \le f \le 1, \quad 0 \le p \le 1, \quad \tau \ge 0, \end{aligned} \tag{5.4}$$

where $\rho = 1$ if X has decreasing hazard rate, and $\rho = 1/2$ otherwise. Unfortunately, in general there are no solutions for this problem due to the strict inequalities. Moreover, since these strict inequalities correspond to stability conditions for the JIQ-subsystem and for the virtual queues in the JOQ-subsystem, a relaxation with non-strict inequalities is not appropriate. Thus, a way to obtain numerical values for the parameters is to introduce some small slack ϵ in the constraints with strict inequalities, so that they become

$$\lambda p \tau \mathbb{E}[\lceil X/\tau \rceil] \le f/2 - \epsilon \qquad \text{and} \qquad \lambda(1-p) \le \rho(1-f) - \epsilon.$$

Furthermore, our simulations (Section 5.5) support the conjecture that the JIQ algorithm is stable for all subcritical arrival rates, regardless of the job size distributions. In that case, the first two constraints (i.e., the stability constraints) in the problem given by (5.4) would need to be

$$\lambda p \tau \mathbb{E}[\lceil X/\tau \rceil] < f \qquad \text{and} \qquad \lambda(1-p) < (1-f). \tag{5.5}$$

For exponential job sizes, we will show that the infimum of the problem given by (5.4) is attained when the parameters are selected as

$$\begin{cases} \text{(i)} & \tau \text{ such that } 1 - e^{-\tau} = 2\lambda\tau, p = 1, f = 1, \\ \text{(ii)} & \tau \text{ s.t. } 2e^{\tau} - 4\tau = 3, p = \frac{(1-\lambda)(e^{\tau}-1)}{\lambda(1-e^{\tau}+2\tau e^{\tau})}, f = 1 - \lambda(1-p), \end{cases} \tag{5.6}$$

when (i) $\lambda \leq \lambda^*$ or (ii) $\lambda > \lambda^*$, where λ^* solves $\frac{(1-\lambda)(e^\tau-1)}{\lambda(1-e^\tau+2\tau e^\tau)} = 1$ when $2e^\tau - 4\tau = 3$. These parameters cause equality in some of the equations where a strict inequality is needed. The solution value is shown in Figure 5.1, as a function of λ. In this figure, we distinguish the cases where JIQ is stable and has vanishing queueing delay for a general job size distribution when $\lambda < 1/2$ (see (5.6)), and where this also holds true for all $\lambda < 1$ (which is widely believed but remains to be proven).

Solution of the minimization problem. The minimization problem (5.4) when job sizes are exponentially distributed looks as follows.

$$\begin{aligned} \inf_{p,f,\tau} \quad & 1 - p + p\frac{e^{-\tau}}{1-e^{-\tau}} \\ \text{s.t.} \quad & \lambda p\tau/(1-e^{-\tau}) < f/2, \quad \lambda(1-p) < 1-f, \\ & 0 \leq f \leq 1, \quad 0 \leq p \leq 1, \quad \tau \geq 0. \end{aligned} \tag{5.7}$$

Note that the objective function is decreasing in τ, so τ should be as large as possible. The first condition then shows that, in order to choose τ large, p should be minimized, after which the second condition gives that f should be chosen as close as possible to but less than $1-(1-p)\lambda$. This transforms the first inequality into

$$\tau/(1-e^{-\tau}) < \frac{1-(1-p)\lambda}{2\lambda p}, \tag{5.8}$$

where the right-hand side is decreasing in p. As we want to maximize τ, we seek equality which gives

$$p(\lambda,\tau) = \frac{(1-\lambda)(e^\tau-1)}{\lambda(1-e^\tau+2\tau e^\tau)}, \tag{5.9}$$

where we note that this value may exceed 1, in which case we preserve the equality as much as possible and set $p(\lambda,\tau) = 1$. Suppose $p(\lambda,\tau) < 1$, in that case we choose τ such that $2e^\tau - 4\tau = 3$ in order to minimize the objective function (this follows from differentiating the objective function and equating it to zero, after substituting (5.9)). However, with this value of τ, $p(\lambda,\tau)$ may be larger than one. Denote this critical value of λ by λ^*, so λ^* is defined as the smallest value of λ^* for which $p(\lambda,\tau)$ is strictly smaller than one, when τ is chosen such that $2e^\tau - 4\tau = 3$. When λ is smaller than this λ^*, we choose $p = 1$ after which the objective function is minimized by putting τ such that (5.8) is met with equality which is the case when $1 - e^{-\tau} = 2\lambda\tau$.

Remark. If the stability result proved in [FS17] applies even when $\lambda < 1$, then (5.8), which corresponds to the first constraint in (5.7), becomes $\lambda p\tau/(1-e^{-\tau}) < f$. The optimal parameters for this problem are

$$\begin{cases} \text{(i)} \quad \tau \text{ such that } 1 - e^{-\tau} = \lambda\tau, p = 1, f = 1, \\ \text{(ii)} \quad \tau \text{ such that } e^{\tau} - 2\tau = 1, p = \frac{(1-\lambda)(e^{\tau}-1)}{\lambda(1-e^{\tau}+\tau e^{\tau})}, f = 1 - \lambda(1-p) \end{cases}$$

when (i) $\lambda \le \lambda^*$ or (ii) $\lambda > \lambda^*$, where λ^* solves $\frac{1-e^{\tau}+\lambda\tau e^{\tau}}{(1-e^{\tau}+\tau e^{\tau})\lambda} = 0$ when $e^{\tau} - 2\tau = 1$.

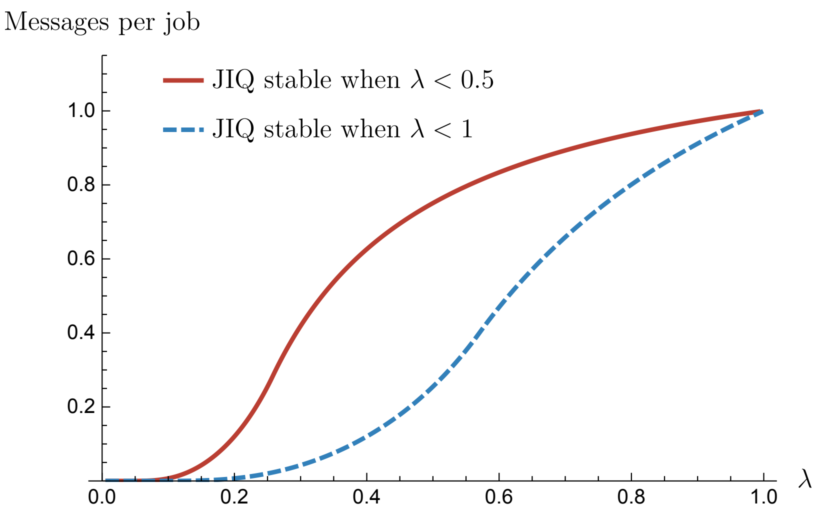
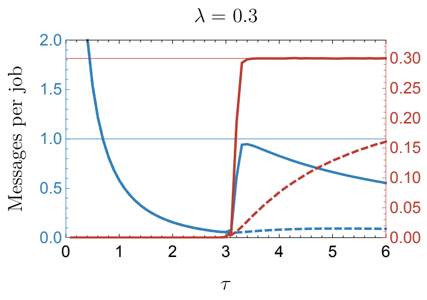


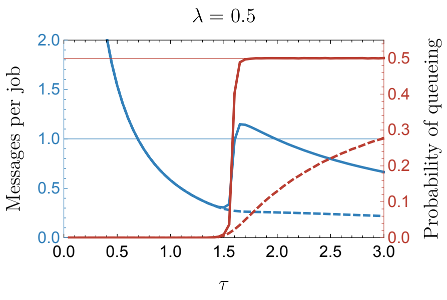
Figure 5.1: Numerical computation: the average number of messages per job as a function of the arrival rate.

5.5 Numerical simulations and discussions

In this section, we performed stochastic simulations of a system with 1000 servers, first with exponential job sizes and later with other job size distributions.



(a) $\lambda = 0.3$.



(b) $\lambda = 0.5$.

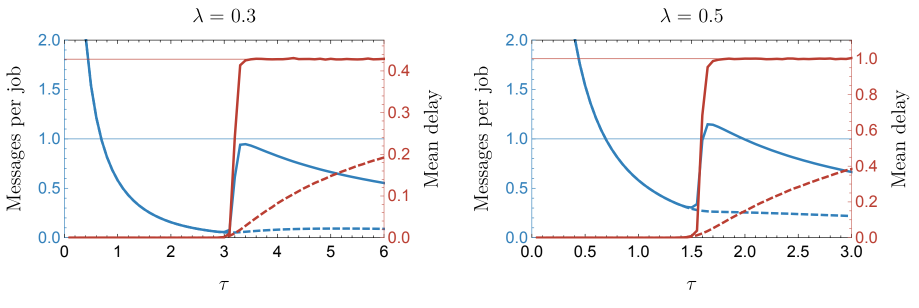
Figure 5.2: Average number of messages per job (left axis) and the probability of queueing (right axis) as a function of τ. The thin blue line represents the number of messages JIQ uses, and the thin red line represents the probability of queueing for the random dispatching policy. The dashed lines represent the performance of the system without virtual queues.

5.5.1 The choice of τ

We investigate the performance of the JOQ algorithm. We therefore plot the probability of queueing as a function of τ for $\lambda = 0.3$ and $\lambda = 0.5$ in Figures Figures 5.2a and 5.2b and the mean delay as a function of τ in Figures 5.3a and 5.3b. We will mainly focus on the probability of queueing, as the observations for the mean delay are almost identical. The dashed lines will become relevant in the next subsection.

In the figures, it is clear that the probability of queueing and the message rate are both extremely low when τ is chosen to be somewhere between 2 and 3. As expected, when τ gets smaller we observe that the average number of messages per job grows, while the probability of queueing is almost zero. This reflects the fact that, as the time between update epochs decreases, the average number of messages increases and the dispatcher has enough open servers to send almost every incoming job to one of them.

Recall that in Section 5.4.2 we described a method to determine the optimal value of τ, as part of the solution of the optimization problem given in (5.4). In particular, for $\lambda = 0.3$ this would lead to $\tau \approx 1.51$. However, in Figure 5.2a, one clearly observes that larger values of τ would give rise to a smaller number of messages, while still yielding essentially zero queueing delay. If we assume that the JIQ algorithm is always stable for any load $\rho < 1$, and use the constraints given in (5.5), then the optimal value of τ would be approximately 3.20 for



(a) $\lambda = 0.3$. (b) $\lambda = 0.5$.

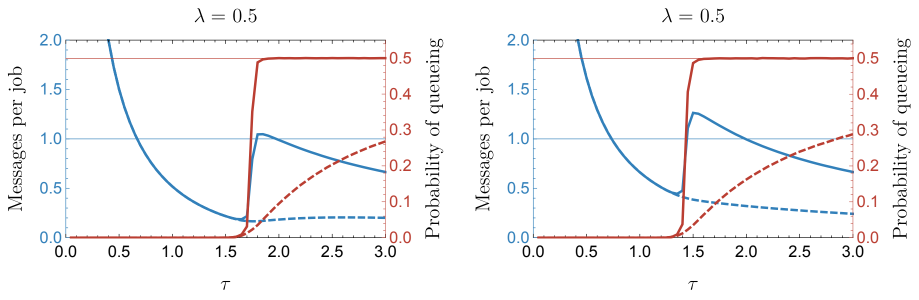
Figure 5.3: Average number of messages per job and the mean delay as a function of τ. The thin blue line represents the number of messages JIQ uses, and the thin red line represents the probability of queueing for the random dispatching policy. The dashed lines represent the performance of the system without virtual queues.

$\lambda = 0.3$ and 1.59 for $\lambda = 0.5$. This matches with the phase transition in the performance observed at this value of τ, where the probability of queueing quickly increases up to the arrival rate λ, and where the average number of messages per job rapidly increases to a local maximum. These sharp increases in both the probability of queueing and the average number of messages per job reflect that when τ is chosen to be too large, the virtual queues become unstable and all servers remain closed. In that case, the dispatcher is forced to send jobs to servers chosen uniformly at random, and servers send messages non-stop every τ time units.

5.5.2 Virtual queues or not

The JOQ algorithm is tractable because we rely on the equivalence with the JIQ algorithm. This is the reason why we use virtual queues instead of the real queues in order to decide for the servers when to stop sending messages.

We compare the JOQ algorithm with the variant without virtual queues, where at an update epoch the server would send a message that it is busy, exactly when it is actually busy. Results are shown in Figures 5.2a, 5.2b, 5.3a and 5.3b, in which the dashed lines represent the results for the JOQ algorithm without virtual queues. As can be seen from these figures, there seems to be no difference for small values of τ. This is because the virtual queues are stable in both cases, as their load is less than one. However, when τ is close to the



(a) Decreasing hazard rate Gamma(2, 2). (b) Increasing hazard rate Gamma(1/2, 1/2).

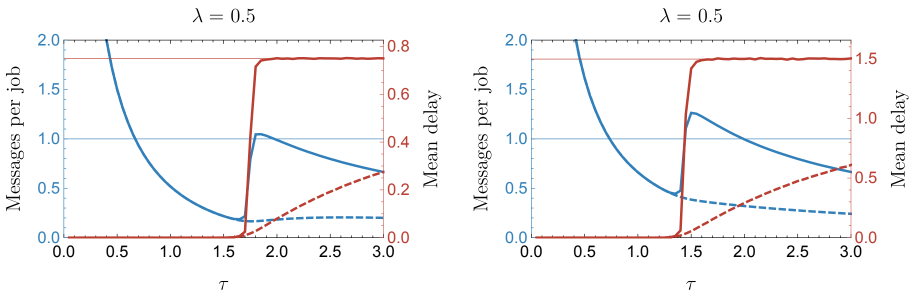
Figure 5.4: Average number of messages per job and the probability of queueing as a function of τ ($\lambda = 0.5$) with non-exponential service times. The dashed lines represent the performance of the system without virtual queues.

crossover point, the difference between having virtual queues or not becomes apparent. The virtual queues become overloaded and no open servers become available anymore, while there are still idle and thus open servers when no virtual queues are used. We do note however that even in the system with no virtual queues, queueing does not seem to vanish if τ is chosen larger than its optimal value.

5.5.3 Non-exponential job size distributions

We chose the update epochs to be equidistant in time for the sake of simplicity of implementation and analysis. While irregular update intervals may reduce the message rate in general, for exponentially distributed job sizes, equidistant update epochs in fact minimize the expected number of messages per job, as will be described further and proven in Section 5.A.

For non-exponential job sizes, the behavior of the mean delay and probability of queueing is quite different, as can be observed in Figures 5.4a, 5.4b, 5.5a and 5.5b. When the hazard rate is decreasing, larger values of τ are acceptable for vanishing delays, which could be explained by observing that (5.2) is an upper bound for the relative load when the job size distribution has decreasing hazard rate, because of (5.1). The mean delay turns out to be smaller in this case as well. On the other hand, when the job size distribution has increasing hazard rate, the mean delay increases and is non-zero already for smaller values of τ. The expected number of messages per job also seems to be larger.



(a) Decreasing hazard rate Gamma(2, 2). (b) Increasing hazard rate Gamma(1/2, 1/2).

Figure 5.5: Average number of messages per job and the mean delay as a function of τ ($\lambda = 0.5$) with non-exponential service times. The dashed lines represent the performance of the system without virtual queues.

5.6 Conclusion

The JOQ algorithm greatly reduces the number of messages exchanged in order to make the probability of queueing and the expected queueing delay vanish, compared to existing algorithms in the literature. In particular, to the best of our knowledge, this is the first algorithm that achieves vanishing queueing delay with less than one message per job in expectation, without advance knowledge of job sizes. Moreover, by mixing JOQ with JIQ, we showed that vanishing queueing delay can be achieved for all systems where JIQ has vanishing queueing delay, but with strictly fewer messages per job on average, compared to JIQ. In fact, for all $\lambda < 1/2$ or any $\lambda < 1$ if the job size distribution has decreasing hazard rate, we use strictly less than one message per job in expectation while achieving a vanishing queueing delay.

It could be worth further investigating how message loss may impact the performance of the JOQ scheme. We believe our scheme does not require any strict timings and can be extended so that it receives messages during specific time windows, instead of at predetermined time epochs. Whenever a message is lost, the dispatcher would only send one message to a potentially busy server. This is better than if a message would get lost in the JIQ scheme, after which one of the servers would be eliminated as option to dispatch to.

An open question is whether the system design 'hybrid JOQ/JIQ' in Section 5.4 minimizes the expected number of messages per job in the limit, for exponentially distributed job sizes. That is, does there exist a system design that also has a vanishing queueing delay, but uses even fewer messages per job in the limit?

A further challenging problem would be to investigate what the optimal scheme looks like for generally distributed job sizes. It seems that a combination of implicit and explicit messaging is necessary.

5.A Optimality of equidistant updates

We will show that equidistant updates are optimal when the service times are exponentially distributed. Let us relax the assumption that, under the JOQ algorithm, the time between update epochs is always the same τ. For $i \geq 1$, let s_i be the time between the $(i-1)$-th and the i-th update epochs, with the convention that the 0-th update epoch is at time 0. Assuming that the probability of queueing converges to 0 as $N \to \infty$, the expected number of messages per job converges to

$$\sum_{i=1}^{\infty} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right).$$

Moreover, the expected size of the blown up jobs is

$$\sum_{i=0}^{\infty} s_{i+1} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right).$$

Assuming that the load of the system is at most ρ, the messaging sequence s^* that minimizes the limit of the expected number of messages per job is the one that solves the following problem.

$$\begin{aligned} \inf_{s \geq 0} \quad & \sum_{i=1}^{\infty} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) \\ s.t. \quad & \lambda \sum_{i=0}^{\infty} s_{i+1} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) \leq \rho. \end{aligned} \tag{5.10}$$

Note that, as long as $\lambda < \rho$, this problem is always feasible. For the special case of exponentially distributed job sizes, we have the following.

Theorem 5.3. *Suppose that the job sizes are exponentially distributed. If s^* attains the infimum in* (5.10)*, then there exists some $\tau > 0$ such that $s_k^* = \tau$, for all $k \geq 1$.*

Proof. We first obtain a sequence of implicit equations for s^*.

Lemma 5.1. *Suppose that $\mathbb{P}(X > x)$ is convex and differentiable, with $f(x) = -\frac{d}{dx}\mathbb{P}(X > x) > 0$, for all $x \geq 0$. Then, there exists a constant $C > 0$ such that*

$$C = \frac{\sum_{i=k}^{\infty} s_{i+1}^* f\left(\sum_{j=1}^{i} s_j^*\right) - \mathbb{P}\left(X > \sum_{j=1}^{k-1} s_j^*\right)}{\sum_{i=k}^{\infty} f\left(\sum_{j=1}^{i} s_j^*\right)} \tag{5.11}$$

for all $k \geq 1$, and

$$C = s_{k+1}^* - \frac{\mathbb{P}\left(\sum_{j=1}^{k-1} s_j^* < X \leq \sum_{j=1}^{k} s_j^*\right)}{f\left(\sum_{j=1}^{k} s_j^*\right)}, \tag{5.12}$$

for all $k \geq 2$.

Proof. Since $\mathbb{P}(X > x)$ is convex, (5.10) is an infinite-dimensional convex optimization problem in s. Now consider the Lagrangian

$$\mathscr{L}(s, y) = \sum_{i=1}^{\infty} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) - y\left[\lambda \sum_{i=0}^{\infty} s_{i+1}\mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) - \rho\right].$$

Let (s^*, y^*) be an optimal solution. For every $k \geq 1$, since $s_k^* = 0$ cannot be optimal, we must have

$$\begin{aligned}\frac{\partial \mathscr{L}(s^*, y^*)}{\partial s_k^*} &= -\sum_{i=k}^{\infty} f\left(\sum_{j=1}^{i} s_j^*\right) \\ &\quad - y^*\lambda\left[-\sum_{i=k}^{\infty} s_{i+1}^* f\left(\sum_{j=1}^{i} s_j^*\right) + \mathbb{P}\left(X > \sum_{j=1}^{k-1} s_j^*\right)\right] = 0.\end{aligned}$$

Since $f(x) > 0$ for all $x \geq 0$, we have $y^* > 0$. Thus, rearranging terms we obtain Equation (5.11) with $C = 1/(y^*\lambda) > 0$. Moreover, using that

$$\frac{\partial \mathscr{L}(s^*, y^*)}{\partial s_k^*} - \frac{\partial \mathscr{L}(s^*, y^*)}{\partial s_{k-1}^*} = 0$$

for all $k \geq 2$, and rearranging terms, we obtain (5.12). $\square$

Combining (5.12) with the fact that the jobs are exponentially distributed, we obtain

$$s_{k+1}^* = C + 1 - e^{s_k^*},$$

for all $k \geq 2$. Combining this with the fact that $s_k^* \geq 0$ for all $k \geq 1$, it can be checked that we must have $s_k^* = \tau$ for all $k \geq 2$, where τ is the unique solution of $\tau = C + 1 - e^{\tau}$.

On the other hand, combining (5.11) for the case $k = 2$, with the fact that the job sizes are exponentially distributed and that $s_k^* = \tau$ for all $k \geq 2$, we obtain

$$\tau = C + \left(1 - e^{\tau}\right) e^{\tau - s_1^*}.$$

Thus, we have $s_1^* = \tau$. $\square$

Bibliography

[AD20] J. Anselmi and F. Dufour. "Power-of-d-Choices with Memory: Fluid Limit and Optimality". In: *Mathematics of Operations Research* 45.3 (2020), pp. 862–888 (cit. on pp. 3, 50, 81).

[AKMOV20] R. Atar, I. Keslassy, G. Mendelson, A. Orda, and S. Vargaftik. "Persistent-Idle Load-Distribution". In: *Stochastic Systems* 10.2 (2020), pp. 152–169 (cit. on p. 4).

[Ans19] J. Anselmi. "Combining Size-Based Load Balancing with Round-Robin for Scalable Low Latency". In: *IEEE Trans. Parallel Distrib. Syst.* (2019) (cit. on p. 12).

[BB08] R. Badonnel and M. Burgess. "Dynamic pull-based load balancing for autonomic servers". In: *Proc. IEEE/IFIP*. 2008, pp. 751–754 (cit. on p. 3).

[BBL17] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. "Load balancing in large-scale systems with multiple dispatchers". In: *Proc. INFOCOM '17*. 2017 (cit. on pp. 11, 15).

[BBL19] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. "Hyper-Scalable JSQ with Sparse Feedback". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3.1 (2019) (cit. on pp. 12, 49).

[BBL20] M. van der Boor, S. C. Borst, and J. S. H. van Leeuwaarden. "Optimal Hyper-Scalable Load Balancing with a Strict Queue Limit". In: *Preprint* (2020) (cit. on pp. 12, 93).

[BBLM18a] M. van der Boor, S. C. Borst, J. S. H. van Leeuwaarden, and D. Mukherjee. “Scalable load balancing in networked systems: A survey of recent advances”. In: *Preprint* (2018) (cit. on p. 10).

[BBLM18b] M. van der Boor, S. C. Borst, J. S. H. van Leeuwaarden, and D. Mukherjee. “Scalable load balancing in networked systems: Universality properties and stochastic coupling methods”. In: *Proc. ICM ’18*. 2018 (cit. on p. 4).

[BCM19] T. Bonald, C. Comte, and F. Mathieu. “Performance of Balanced Fairness in Resource Pools: A Recursive Approach”. In: *ACM SIGMETRICS Performance Evaluation Review* 46 (2019), pp. 125–127 (cit. on p. 4).

[BD11] J. R. Boucherie and N. van Dijk. *Queueing Netwoks: A Fundamental Approach*. Wiley, 2011 (cit. on p. 8).

[BKK95] S. Berezner, C. Kriel, and A. Krzesinski. “Quasi-reversible multiclass queues with order independent departure rates”. In: *Queueing Systems* 19 (1995), pp. 345–359 (cit. on p. 104).

[BLP10] M. Bramson, Y. Lu, and B. Prabhakar. “Randomized load balancing with general service time distributions”. In: *Proc. SIGMETRICS ’10*. 2010, pp. 275–286 (cit. on p. 2).

[BLP12] M. Bramson, Y. Lu, and B. Prabhakar. “Asymptotic independence of queues under randomized load balancing”. In: *Queueing Syst.* 71.3 (2012), pp. 247–292 (cit. on p. 2).

[BZB20] M. van der Boor, M. Zubeldia, and S. C. Borst. “Zero-wait load balancing with sparse messaging”. In: *Operations Research Letters* 48.3 (2020), pp. 368–375 (cit. on pp. 13, 123).

[Cec18] F. Cecchi. “Mean-field limits for ultra-dense random-access networks”. English. PhD thesis. Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 2018 (cit. on p. 10).

[Com19] C. Comte. “Dynamic load balancing with tokens”. In: *Computer Communications* 144 (2019), pp. 76 –88 (cit. on p. 4).

[DLMF] *NIST Digital Library of Mathematical Functions*. dlmf.nist.gov/, Release 1.0.28 of 2020-09-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. (cit. on p. 47).

[EG18] P. Eschenfeldt and D. Gamarnik. “Join the Shortest Queue with Many Servers. The Heavy-Traffic Asymptotics”. In: *Mathematics of Operations Research* 43.3 (2018), pp. 867–886 (cit. on p. 2).

[EVW80] A. Ephremides, P. Varaiya, and J. Walrand. “A simple dynamic routing problem”. In: *IEEE Trans. Autom. Control* 25.4 (1980), pp. 690–693 (cit. on p. 1).

[FS17] S. G. Foss and A. L. Stolyar. “Large-scale Join-Idle-Queue system with general service times”. In: *J. Appl. Probab.* 54.4 (2017), pp. 995–1007 (cit. on pp. 3, 127, 128, 130, 131, 134).

[Gan+14] R Gandhi et al. “Duet: Cloud scale load balancing with hardware and software”. In: *ACM SIGCOMM Computer Communication Review* 44.4 (2014), pp. 27–38 (cit. on p. 5).

[Gas17] N. Gast. “Expected values estimated via mean-field approximation are 1/N-accurate”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.1 (2017), p. 17 (cit. on p. 58).

[GTZ16] D. Gamarnik, J. Tsitsiklis, and M. Zubeldia. “Delay, memory and messaging tradeoffs in distributed service systems”. In: *Proc. SIGMETRICS ’16*. 2016, pp. 1–12 (cit. on pp. 2, 5).

[GW19] V. Gupta and N. Walton. “Load Balancing in the Nondegenerate Slowdown Regime”. In: *Oper. Res.* 67.1 (2019), pp. 281–294 (cit. on p. 2).

[HK94] P. Hunt and T. Kurtz. “Large loss networks”. In: *Stoch. Proc. Appl.* 53.2 (1994), pp. 363–378 (cit. on pp. 9, 21, 30, 56, 71, 72).

[HV20] T. Hellemans and B. Van Houdt. “Performance Analysis of Load Balancing Policies with Memory”. In: *Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS ’20. Tsukuba, Japan, 2020, 27–34 (cit. on p. 3).

[Jag74] D. Jagerman. “Some properties of the Erlang loss function”. In: *The Bell System Technical Journal* 53.3 (1974), pp. 525–551 (cit. on pp. 47, 48).

[JLZ08] A. J.E. M. Janssen, J. S. H. van Leeuwaarden, and B. Zwart. “Gaussian expansions and bounds for the Poisson distribution applied to the Erlang B formula”. In: *Advances in Applied Probability* 40.1 (2008), 122–143 (cit. on p. 48).

[Kel11] F. P. Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011, pp. 6–7 (cit. on pp. 8, 22, 104).

[Krz11] A. Krzesinski. "Order Independent Queues". In: *R. Boucherie , N. van Dijk (eds) Queueing Networks. International Series in Operations Research and Management Science*. Vol. 154. 2011 (cit. on pp. 104, 116).

[LT09] J. S. H. van Leeuwaarden and N. M. Temme. "Asymptotic inversion of the Erlang B formula". In: *SIAM Journal on Applied Mathematics* 70.1 (2009), pp. 1–23 (cit. on p. 48).

[Lu+11] Y. Lu et al. "Join-idle-queue: a novel load balancing algorithm for dynamically scalable web services". In: *Perform. Eval.* 68.11 (2011), pp. 1056–1071 (cit. on pp. 3, 4).

[LY18] X. Liu and L. Ying. "On Achieving Zero Delay with Power-of-d-Choices Load Balancing". In: *2018 IEEE Conference on Computer Communications (INFOCOM)*. 2018, pp. 297–305 (cit. on p. 3).

[MBLW16a] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, and P. Whiting. "Asymptotic Optimality of Power-of-d Load Balancing in Large-Scale Systems". In: *Math. Oper. Res* (2016) (cit. on p. 3).

[MBLW16b] D. Mukherjee, S. C. Borst, J. S. H. van Leeuwaarden, and P. A. Whiting. "Universality of load balancing schemes on the diffusion scale". In: *J. Appl. Probab.* 53.4 (2016) (cit. on p. 4).

[Mit01] M. Mitzenmacher. "The power of two choices in randomized load balancing". In: *IEEE Trans. Parallel Distrib. Syst.* 12.10 (2001), pp. 1094–1104 (cit. on p. 2).

[Mit16] M. Mitzenmacher. "Analyzing distributed Join-Idle-Queue: A fluid limit approach". In: *Proc. Allerton 2016*. 2016, pp. 312–318 (cit. on p. 4).

[MKMG15] A. Mukhopadhyay, A. Karthik, R. R. Mazumdar, and F. Guillemin. "Mean field and propagation of chaos in multi-class heterogeneous loss models". In: *Perform. Eval.* 91 (2015), pp. 117–131 (cit. on p. 2).

[MSY12] S. T. Maguluri, R Srikant, and L Ying. "Stochastic models of load balancing and scheduling in cloud computing clusters". In: *INFOCOM, 2012 Proceedings IEEE*. IEEE. 2012, pp. 702–710 (cit. on p. 5).

[Muk18] D. Mukherjee. “Scalable load balancing algorithms in networked systems”. English. PhD thesis. Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 2018 (cit. on p. 10).

[Pat+13] P. Patel et al. “Ananta: cloud scale load balancing”. In: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), pp. 207–218 (cit. on p. 5).

[PTW07] G. Pang, R. Talreja, and W. Whitt. “Martingale proofs of many-server heavy-traffic limits for Markovian queues”. In: *Probability Surveys* 4 (2007), pp. 193–267 (cit. on pp. 9, 21, 30, 56, 71, 72).

[Sto15] A. L. Stolyar. “Pull-based load distribution in large-scale heterogeneous service systems”. In: *Queueing Syst.* 80.4 (2015), pp. 341–361 (cit. on pp. 3, 131).

[Sto17] A. L. Stolyar. “Pull-based load distribution among heterogeneous parallel servers: the case of multiple routers”. In: *Queueing Syst.* 85.1 (2017), pp. 31–65 (cit. on pp. 4, 20).

[Tem16] N. Temme. *Personal communication*. 2016 (cit. on p. 47).

[TX12] J. N. Tsitsiklis and K Xu. “On the power of (even a little) resource pooling”. In: *Stochastic Systems* 2.1 (2012), pp. 1–66 (cit. on p. 79).

[TX13] J. N. Tsitsiklis and K. Xu. “Queueing system topologies with limited flexibility”. In: *Proc. SIGMETRICS ’13*. 2013 (cit. on p. 79).

[VDK96] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich. “Queueing system with selection of the shortest of two queues: An asymptotic approach”. In: *Problemy Peredachi Informatsii* 32.1 (1996), pp. 20–34 (cit. on p. 2).

[VKO20] S. Vargaftik, I. Keslassy, and A. Orda. “LSQ: Load Balancing in Large-Scale Heterogeneous Systems With Multiple Dispatchers”. In: *IEEE/ACM Transactions on Networking* (2020), pp. 1–13 (cit. on p. 4).

[Win77] W. Winston. “Optimality of the shortest line discipline”. In: *J. Appl. Probab.* 14.1 (1977), pp. 181–189 (cit. on p. 1).

[WW20] W. Weng and W. Wang. “Achieving Zero Asymptotic Queueing Delay for Parallel Jobs”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 4.3 (2020) (cit. on p. 3).

[XDLS15] Q. Xie, X. Dong, Y. Lu, and R. Srikant. “Power of d choices for large-scale bin packing”. In: *Proc. SIGMETRICS ’15*. 2015, pp. 321–334 (cit. on p. 2).

[Yin16] L. Ying. “On the approximation error of mean-field models”. In: *Proc. SIGMETRICS 2016/Performance 2016*. ACM Press, 2016, pp. 285–297 (cit. on p. 58).

[YSK15] L. Ying, R. Srikant, and X. Kang. “The power of slightly more than one sample in randomized load balancing”. In: *Proc. INFOCOM 2015*. 2015, pp. 1131–1139 (cit. on p. 2).

[ZSW20] X. Zhou, N. Shroff, and A. Wierman. “Asymptotically optimal load balancing in large-scale heterogeneous systems with multiple dispatchers”. In: *Performance Evaluation* (2020) (cit. on pp. 4, 5).

Summary

Scalable load balancing algorithms (LBAs) achieve excellent delay performance in large-scale systems and yet only involve low implementation overhead. LBAs play a critical role in distributing service requests or tasks (e.g. compute jobs, data base lookups, file transfers) among servers or distributed resources in parallel-processing systems. The analysis and design of LBAs has attracted strong attention in recent years, mainly spurred by crucial scalability challenges arising in cloud networks and data centers with massive numbers of servers handling a huge influx of service requests. The use of state information naturally allows dynamic LBAs to achieve better delay performance, but also involves higher implementation complexity and a substantial communication burden.

Motivated by these issues, we introduce and analyze novel scalable LBAs which strike an optimal trade-off between performance and communication overhead. Specifically, LBAs are referred to as 'hyper-scalable' when they operate below the minimum requirement for vanishing delay in a many-server regime, referred to as the hyper-scalable operating region. Mathematical techniques such as queueing networks and fluid limits will be used for the analysis of these LBAs, and extensive simulation experiments are conducted to illustrate the results.

In Chapter 2, we consider a multiple-dispatcher scenario where the loads may differ among dispatchers. We leverage product-form representations and fluid limits to establish that the blocking and wait then no longer vanish when applying JIQ, even for arbitrarily low overall load. Remarkably, it is the least-loaded dispatcher that throttles tokens and leaves idle servers stranded, thus acting as bottleneck. We introduce two enhancements of the ordinary JIQ scheme where tokens are either distributed non-uniformly or occasionally exchanged among the various dispatchers. We prove that these extensions can achieve zero blocking and wait in the many-server limit, for any subcritical over-

all load and arbitrarily skewed load profiles.

In Chapter 3, we consider a hyper-scalable scheme where the various servers provide occasional queue updates to guide the load assignment. We show that the proposed schemes can achieve a vanishing delay in the many-server limit with just one message per job, just like the popular Join-the-Idle-Queue (JIQ) scheme. We investigate fluid limits for synchronous updates as well as asynchronous exponential update intervals. The fixed point of the fluid limit is identified in the latter case and used to derive the queue length distribution. We also demonstrate that in the ultra-low feedback regime the mean stationary delay tends to a constant in the synchronous case, but grows without bound in the asynchronous case.

In Chapter 4, we analyze a scenario in which jobs may only be admitted when a specific limit on the queue position of the job can be guaranteed. The centerpiece of our analysis is a universal upper bound for the achievable throughput of any dispatcher-driven algorithm for a given communication budget and queue limit. We also propose a specific hyper-scalable scheme which can operate at any given message rate and enforce any given queue limit, while allowing the server states to be captured via a closed product-form network, in which servers act as customers traversing various nodes. The bound is tight and the proposed hyper-scalable scheme is throughput-optimal in a many-server regime given the communication and queue limit constraints.

Finally, in Chapter 5, we introduce a novel hyper-scalable scheme in which the dispatcher becomes aware of idle servers without any explicit communication from either side, using absence of messages at predefined time instants. The proposed scheme achieves provably vanishing queueing delays while using strictly less than one message per job on average.

About the author

Mark van der Boor was born in Sliedrecht, The Netherlands, on August 22, 1993. After finishing VWO in 2011 at Het Willem de Zwijger College in Papendrecht, he studied Industrial and Applied Mathematics at Eindhoven University of Technology (TU/e). In 2014, he received his Bachelor's degree with honor (cum laude). He then pursued a Master's degree in Applied Mathematics at TU/e, specializing in Statistics, Probability and Operations Research, where he wrote a Master's thesis under the supervision of Sem Borst and Johan van Leeuwaarden, and obtained his degree with honor (cum laude) in 2016.

In November 2016, Mark started a PhD project at TU/e in the department of Mathematics and Computer Science, once again under the supervision of Sem Borst and Johan van Leeuwaarden, of which the results are presented in this dissertation. The PhD project was part of the Dutch research consortium NETWORKS.

Mark has worked at RTV Papendrecht since 2011. During his PhD, Mark has been involved in several teaching activities, such as lecturing Stochastic Simulation and Stochastics and Simulation for Finance. He had an internship at IBM Yorktown in 2018 and co-organized the 2019 edition of the Young European Queueing Theorists (YEQT) workshop.

Mark will defend his PhD thesis on March 26, 2021.