http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–16
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Self-Learning Threshold-Based Load Balancing

Diego Goldsztajn,[a] Sem C. Borst,[a] Johan S. H. van Leeuwaarden,[b] Debankur Mukherjee,[c] Philip A. Whiting[d]

[a] Eindhoven University of Technology, Eindhoven 5612 AZ, Netherlands; [b] Tilburg University, Tilburg 5037 AB, Netherlands; [c] Georgia Institute of Technology, Atlanta, Georgia 30332; [d] Macquarie University, Macquarie Park, New South Wales 2109, Australia

Contacts: d.e.goldsztajn@tue.nl, https://orcid.org/0000-0001-7212-0309 (DG); s.c.borst@tue.nl (SCB); j.s.h.vanleeuwaarden@uvt.nl (JSHvL); debankur.mukherjee@isye.gatech.edu, https://orcid.org/0000-0003-1678-4893 (DM); philip.whiting@mq.edu.au (PAW)

Received: November 13, 2020
Revised: April 1, 2021; May 10, 2021
Accepted: May 12, 2021
Published Online in Articles in Advance: September 16, 2021

https://doi.org/10.1287/ijoc.2021.1100

Abstract. We consider a large-scale service system where incoming tasks have to be instantaneously dispatched to one out of many parallel server pools. The user-perceived performance degrades with the number of concurrent tasks and the dispatcher aims at maximizing the overall quality of service by balancing the load through a simple threshold policy. We demonstrate that such a policy is optimal on the fluid and diffusion scales, while only involving a small communication overhead, which is crucial for large-scale deployments. In order to set the threshold optimally, it is important, however, to learn the load of the system, which may be unknown. For that purpose, we design a control rule for tuning the threshold in an online manner. We derive conditions that guarantee that this adaptive threshold settles at the optimal value, along with estimates for the time until this happens. In addition, we provide numerical experiments that support the theoretical results and further indicate that our policy copes effectively with time-varying demand patterns.

Summary of Contribution: Data centers and cloud computing platforms are the digital factories of the world, and managing resources and workloads in these systems involves operations research challenges of an unprecedented scale. Due to the massive size, complex dynamics, and wide range of time scales, the design and implementation of optimal resource-allocation strategies is prohibitively demanding from a computation and communication perspective. These resource-allocation strategies are essential for certain interactive applications, for which the available computing resources need to be distributed optimally among users in order to provide the best overall experienced performance. This is the subject of the present article, which considers the problem of distributing tasks among the various server pools of a large-scale service system, with the objective of optimizing the overall quality of service provided to users. A solution to this load-balancing problem cannot rely on maintaining complete state information at the gateway of the system, since this is computationally unfeasible, due to the magnitude and complexity of modern data centers and cloud computing platforms. Therefore, we examine a computationally light load-balancing algorithm that is yet asymptotically optimal in a regime where the size of the system approaches infinity. The analysis is based on a Markovian stochastic model, which is studied through fluid and diffusion limits in the aforementioned large-scale regime. The article analyzes the load-balancing algorithm theoretically and provides numerical experiments that support and extend the theoretical results.

History: Accepted by Nicola Secomandi, Area Editor for Stochastic Models and Reinforcement Learning.

Funding: This work was supported by the Netherlands Organisation for Scientific Research (NWO) [Gravitation Grant NETWORKS-024.002.003], by the Australian Research Council [Discovery Project Grant DP180103550], and the National Science Foundation (NSF) [Grant 2113027].

Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2021.1100.

Keywords: adaptive load balancing • many-server asymptotics • fluid and diffusion limits

1. Introduction

Consider a service system where incoming tasks have to be immediately routed to one out of many parallel server pools. The service of tasks starts upon arrival and is independent of the number of tasks contending for service at the same server pool. Nevertheless, the portion of shared resources available to individual tasks does depend on the number of contending tasks, and, in particular, the experienced performance may degrade as the degree of contention rises, creating an incentive to balance the load so as to keep the maximum number of concurrent tasks across server pools as low as possible.

The latter features are characteristic of video streaming applications, such as video conferencing services. In this context, a server pool could correspond to an individual

server, instantiated to handle multiple streaming tasks in parallel. The duration of tasks is typically determined by the application and is not significantly affected by the number of instances contending for the finite shared resources of the server (e.g., bandwidth). However, the video and audio quality suffer degradation as these resources get distributed among a growing number of active instances. Effective load-balancing policies are thus key to optimizing the overall user experience, but the implementation of these policies must be simple enough as to not introduce significant overheads, particularly in large systems.

As it is standard in the load-balancing literature, suppose that the execution times of tasks are exponentially distributed, with unit mean, and that tasks arrive as a Poisson process with aggregate intensity equal to the arrival rate scaled by the number of server pools. The number of tasks in steady state is Poisson distributed with mean equal to this intensity. Thus, a natural and simple load-balancing strategy is a threshold policy that assigns incoming tasks to server pools that exhibit a number of tasks smaller than the rounded-up arrival rate, if possible. In fact, we prove that a policy of this kind is optimal on the fluid scale: in large-scale systems, the fraction of server pools that feature a number of tasks that exceeds or falls short of the rounded-up or rounded-down arrival rate vanishes over time; we further show that this policy is still optimal on the more fine-grained diffusion scale and only involves a small communication overhead. To achieve optimality, however, the threshold must be learned, since it depends on the overall demand for service, which may be unknown or even time-varying. For this purpose, we introduce a control rule for adjusting the threshold in an online fashion, relying solely on the same state information needed to take the dispatching decisions.

Effectively, our adaptive dispatching rule integrates online resource-allocation decisions with demand estimation. Whereas these two attributes are evidently intertwined, the online control actions and longer-term estimation rules are typically decoupled and separately studied in the literature. The former usually assume perfect knowledge of relevant system parameters, whereas the latter typically focus on statistical estimation of these parameters. In contrast, our policy smoothly blends these two elements and does not rely on an explicit load estimate; instead, it yields an implicit indication as a by-product.

We analyze this policy theoretically, through fluid and diffusion approximations that are justified by suitable large-scale limit theorems, and also by means of several numerical experiments; our main contributions are as follows:

- We establish that a threshold policy is optimal on the fluid and the diffusion scales if the threshold is chosen suitably. In addition, we provide a token-based implementation that involves a low communication overhead of at most two messages per task.
- The optimal threshold depends on the offered load, which tends to be uncertain or even time-varying in practice. We propose a control rule for adjusting the threshold in an online manner to an unknown load, relying solely on the tokens kept at the dispatcher. To analyze this rule, we provide a fluid limit for the joint evolution of the system occupancy and the self-learning threshold. To the best of our knowledge, this is the first paper that tackles control parameter adaptations of this kind as part of a fluid-limit analysis.
- We prove that the threshold settles in finite time in a many-server regime, and we provide lower and upper bounds for the equilibrium threshold. These are used to design the control rule to achieve nearly-optimal performance once the threshold has reached an equilibrium. In addition, we derive an upper bound for the limit of the time until the threshold settles as the number of tasks grows large.
- The theoretical results are accompanied by several numerical experiments, which show that the threshold settles in systems with a few hundred servers and only after a short time. Furthermore, even in the presence of highly variable demand patterns, our simulations indicate that the threshold adapts swiftly to variations in the offered load.

Load-balancing problems, similar to the one addressed in the present paper, have received immense attention in the past few decades; van der Boor et al. (2018) provide a recent survey. Whereas traditionally the focus in this literature used to be on performance, more recently the implementation overhead has emerged as an equally important issue. This overhead has two sources: the communication burden of exchanging messages between the dispatcher and the servers, and the cost in large-scale deployments of storing and managing state information at the dispatcher, as considered in Gamarnik et al. (2018).

Although this paper concerns an "infinite-server" setting, the load-balancing literature is predominantly focused on single-server models, where performance is generally measured in terms of queue lengths or delays. In that scenario, the join-the-shortest-queue (JSQ) policy minimizes the mean delay for exponentially distributed service times, among all nonanticipating policies; see Winston (1977) and Ephremides et al. (1980). However, a naive implementation of this policy involves an excessive communication burden for large systems. So-called power-of-d strategies assign tasks to the shortest among d randomly sampled queues, which involves substantially less communication overhead and yet provides significant improvements in delay performance over purely random routing, even for $d = 2$; see Vvedenskaya et al. (1996), Mitzenmacher (2001), and Mukherjee et al. (2016). A further alternative are

pull-based policies, which were introduced in Badonnel and Burgess (2008) and Stolyar (2015). These policies reduce the communication burden by maintaining state information at the dispatcher. In particular, the join-the-idle-queue (JIQ) policy studied in Lu et al. (2011) and Stolyar (2015) matches the optimality of JSQ in a many-server regime and involves only one message per task. This is achieved by storing little state information at the dispatcher, in the form of a list of idle queues.

The main differences in the delay performance of the aforementioned policies appear in the heavy-traffic regime where the load approaches one. If we take JSQ as a reference, then JIQ deviates from this benchmark under certain heavy-traffic conditions. This issue was addressed in Zhou et al. (2017, 2019), which propose join below the threshold (JBT): a refinement of JIQ for achieving heavy-traffic delay optimality at the expense of increasing the communication overhead only mildly. Despite similarity in name, the problem considered in the latter papers is essentially different from the one addressed in the present paper, since achieving delay optimality in a system of parallel single-server queues does not require to maintain a balanced distribution of the load; in fact, the JBT policy does not maintain even queue lengths. The JBT policy was considered for systems of heterogeneous limited processor-sharing servers with state-dependent service rates in Horváth et al. (2019). Individual limited processor-sharing systems with state-dependent service rates were studied by Gupta and Harchol-Balter (2009), who analyze how to set the multiprogramming limit to minimize the mean response time in a way that is robust with respect to the arrival process; this is a scheduling problem where the way in which the service rate changes with the number of tasks sharing the server is a crucial factor. In the context of purely processor-sharing servers with finite buffers, dispatching policies that are insensitive to the job size distribution, as in Bonald et al. (2004), have received considerable attention. Jonckheere and Prabhu (2016) study the asymptotic loss probability of an insensitive dispatching policy in a symmetric scenario, and Comte (2019) proposes a token-based insensitive load-balancing rule for a system with different classes of both jobs and servers, assuming balanced service rates across the server classes.

As we have mentioned, the infinite-server setting considered in this paper has received only limited attention in the load-balancing literature. Whereas queue lengths and delays are hardly meaningful in this type of scenario, load balancing still plays a crucial role in optimizing different performance measures, and many of the concepts discussed in the single-server context carry over. One relevant performance measure is the loss probability in Erlang-B scenarios. Power-of-d properties for these probabilities have been established in Turner (1998), Mukhopadhyay et al. (2015a,b), Xie et al. (2015), and Karthik et al. (2017). Other relevant measures are Schur-concave utility metrics associated with quality of service as perceived by streaming applications; these metrics are maximized by balancing the load. As in the single-server setting, JSQ is the optimal policy for evenly distributing tasks among server pools, but it involves a significant implementation burden; see Menich and Serfozo (1991) and Sparaggis et al. (1993) for proofs of the optimality of JSQ. It was established in Mukherjee et al. (2020) that the performance of JSQ can be asymptotically matched by certain power-of-d strategies that reduce the communication overhead significantly, by sampling a suitably chosen number of server pools that depends on the number of tasks and dispatching tasks to the least congested of the sampled server pools.

Just like the strategies studied in Mukherjee et al. (2020), the policy considered in this paper aims at optimizing the overall experienced performance and asymptotically matches the optimality of JSQ on the fluid and diffusion scales. Moreover, our policy involves at most two messages per task and requires to store only two tokens per server pool at the dispatcher; from this perspective, our policy is the counterpart of JIQ in the infinite-server setting. Another pull-based strategy in an infinite-server (blocking) scenario was briefly considered in Stolyar (2015). Whereas this policy minimizes the loss probability, it does not achieve an even distribution of the load and involves storing a larger number of tokens: one for each idle server in the system. From a technical perspective, we use a similar methodology to derive a fluid limit in the case of a static threshold. However, a different proof method is needed when the threshold is adjusted over time.

The infinite-server model considered in this paper represents a scenario with streaming applications that have random but fixed session durations and adaptive resource requirements (e.g., in terms of bandwidth) in the presence of a time-varying number of competing flows. This setup has been commonly adopted in the literature as a natural modeling paradigm for describing the dynamics and evaluating the performance of streaming sessions on flow-level; see, for instance, Benameur et al. (2002) and Key et al. (2004).

As alluded to earlier, the most appealing feature of our policy is its capability of adapting the threshold to unknown and time-varying loads. The problem of adaptation to uncertain demand patterns was addressed in the single-server setting in Mukherjee et al. (2017) and Goldsztajn et al. (2018a,b), which remove the classical fixed-capacity assumption of the single-server load-balancing literature and assume instead that the number of servers can be adjusted on the fly to match the load. However, in these papers, the load-balancing policy remains the same at all times, since the "right-sizing" mechanism for adjusting the capacity of the system is sufficient to deal with changes in demand. Mechanisms of this kind had already been

studied in the single-server scenario to trade off latency and power consumption in microprocessors; see Yao et al. (1995) and Wierman et al. (2012). As in the present paper, all of the aforementioned papers attempt to learn the load of the system by observing certain state information; a treatment of the more general problem of estimating parameters of a queueing system by observing its trajectories can be found in Baccelli et al. (2009). A different option is to forecast demand from historical data, which has been the predominant approach within the call center staffing literature; a recent survey of this literature is provided in Defraeye and van Nieuwenhuyse (2016).

Different from any of the aforementioned studies, this paper considers a pull-based dispatching policy for optimizing the overall quality of service in a system of parallel server pools. This policy is endowed with a self-learning capability that seamlessly adapts to unknown load values in an online manner and additionally tracks load variations that are prevalent in practice but rarely accounted for in the load-balancing literature.

The remainder of the paper is organized as follows. In Section 2, we describe the model and our dispatching policy. In Section 3, we establish that this policy is fluid and diffusion-optimal if the threshold is chosen suitably. In Section 4, we analyze a control rule for adjusting the threshold to an unknown load, and we explain how to tune this control rule for nearly-optimal performance. Our theoretical results are corroborated by numerical experiments[1] in Section 5, and conclusions are provided in Section 6. The proofs of some technical results are provided in the online supplement.

2. Model and Threshold Policy

In this section, we specify a stochastic model for a service system of parallel and identical server pools, and we describe a threshold-based load-balancing policy for assigning the incoming tasks to the server pools. The model description is carried out in Section 2.1, and the load-balancing policy is explained in Section 2.2.

2.1. Model Description

Consider n parallel and identical server pools with infinitely many servers each. Tasks arrive as a Poisson process with rate $n\lambda$, where λ is the individual server-pool arrival rate, and are immediately routed to one of the server pools, where service starts at once and lasts an exponentially distributed time of unit mean. The execution times are independent of the number of tasks contending for service at the same server pool, but the quality of service experienced by tasks degrades as the degree of contention increases. Thus, maintaining an even distribution of the load is key to optimizing the overall quality of service.

More specifically, let X_i denote the number of concurrent tasks at server pool i, and suppose that we resort to a utility metric $u(X_i) = g(1/X_i)$ as a proxy for measuring the quality of service experienced by a task assigned to server pool i, as a function of its resource share. Provided that g is a concave and increasing function, the overall utility $\sum_{i=1}^{n} X_i u(X_i)$ is a Schur-concave function of $X = (X_1, \ldots, X_n)$ and is thus maximized by balancing the number of tasks among the various server pools.

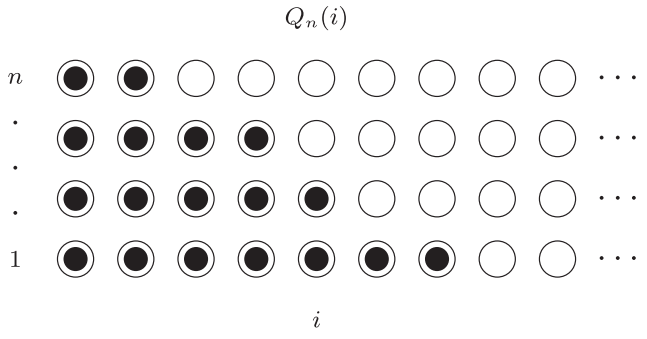
The vector-valued process X describing the number of tasks at each of the n server pools constitutes a continuous-time Markov chain when the dispatching decisions are based on the current number of tasks at each server pool. It is, however, more convenient to adopt an aggregate state description, denoting by $Q_n(i)$ the number of server pools with at least i tasks, as illustrated in Figure 1. In view of the symmetry of the model, the infinite-dimensional process $Q_n = \{Q_n(i) : i \geq 0\}$ also constitutes a continuous-time Markov chain. We will often consider the normalized processes $q_n(i) = Q_n(i)/n$ and $q_n = \{q_n(i) : i \geq 0\}$; the former corresponds to the fraction of server pools with at least i tasks.

2.2. Threshold-Based Load-Balancing Policy

All server pools together constitute an infinite-server system, and thus the total number of tasks in stationarity is Poisson distributed with mean $n\lambda$. In order to motivate our dispatching rule, let us briefly assume that $n\lambda \in \mathbb{N}$. If at a given time the total number of tasks in the system was exactly $n\lambda$, then the diagram of Figure 1 would ideally consist of rows with either $\lfloor \lambda \rfloor$ or $\lceil \lambda \rceil$ tasks, where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions, respectively. This load distribution corresponds to the sequence q^* defined as

$$q^*(i) = 1 \quad \text{if} \quad i \leq \lfloor \lambda \rfloor, \quad q^*(\lfloor \lambda \rfloor + 1) = \lambda - \lfloor \lambda \rfloor \quad \text{and} \quad q^*(i) = 0 \quad \text{if} \quad i > \lfloor \lambda \rfloor + 1. \tag{1}$$

Figure 1. Schematic Representation of the System State



Notes. White circles represent servers, and black circles represent tasks. Each row corresponds to a server pool, and these are arranged so that the number of tasks increases from top to bottom. The number of tasks in column i equals $Q_n(i)$.

When the dispatching rule is JSQ, it is shown in Mukherjee et al. (2020) that q_n has a stationary distribution for each n and that these stationary distributions converge to the Dirac probability measure concentrated at q^* as $n \to \infty$. However, this comes at the expense of a significant communication burden, as observed in Section 1.

The total number of tasks fluctuates over time, but if tasks are dispatched in a suitable way, then it is possible that all server pools have either $\lfloor \lambda \rfloor$ or $\lceil \lambda \rceil$ tasks most of the time and that only the fraction of server pools with each of these occupancy levels fluctuates. To achieve this, we propose a dispatching rule based on a threshold $\ell \in \mathbb{N}$ and the current system occupancy; for brevity, we also define $h = \ell + 1$. This policy is as follows:

- If $q(\ell) < 1$, then some server pool has strictly less than ℓ tasks. In this case, new tasks are assigned to a server pool with strictly less than ℓ tasks, chosen uniformly at random.
- If $q(\ell) = 1$ and $q(h) < 1$, then all server pools have at least ℓ tasks, and at least one server pool has exactly ℓ tasks. In this case, incoming tasks are sent to a server pool chosen uniformly at random among those with exactly ℓ tasks.
- If $q(h) = 1$, then all server pools have more than ℓ tasks. In this case, new tasks are assigned to a server pool with more than ℓ tasks, chosen uniformly at random.

At this point, the main question is if there exists a threshold for which this policy results in an even distribution of the load. Before addressing this question, though, we propose a token-based implementation of this policy, which involves a small communication overhead.

In this implementation, the dispatcher stores at most two tokens per server pool, labeled "green" and "yellow." A server pool has a green token at the dispatcher if it has strictly less than ℓ tasks, and it has a yellow token if it has strictly less than h tasks; note that a server pool with strictly less than ℓ tasks will have both a green and a yellow token. When a task arrives to the system, the dispatcher uses these green and yellow tokens as follows:

- In the presence of green tokens, the dispatcher picks one uniformly at random and sends the task to the corresponding server pool; the token is then discarded.
- If the dispatcher only has yellow tokens, then one of these tokens is chosen uniformly at random and the task is sent to the corresponding server pool; the token is then discarded.
- In the absence of tokens, the task is sent to a server pool chosen uniformly at random.

In order to maintain accurate state information at the dispatcher, the server pools send messages with updates about their status. A server pool with exactly h tasks that finishes one of its tasks will send a yellow message to the dispatcher in order to generate a yellow token. Similarly, a server pool with exactly ℓ tasks that finishes one of these tasks will send a green message to the dispatcher to generate a green token. In addition, green messages are also triggered by arrivals when the number of tasks in the server pool receiving the new task is still strictly less than ℓ after the arrival; in this way, the green token discarded by the dispatcher is replaced.

With this implementation, a given task may trigger at most two messages: one upon arrival and one after leaving the system; that is, the communication overhead is at most two messages per task. Also, note that the amount of memory needed at the dispatcher corresponds to $2n$ tokens. In the next section, we will establish that our policy is optimal on the fluid and diffusion scales for a suitable threshold. This powerful combination of optimality and low communication overhead resembles the properties of JIQ, as considered in the context of single-server queues; see Lu et al. (2011) and Stolyar (2015).

3. Optimality Analysis

In this section, we prove that our policy is optimal on the fluid and diffusion scales, provided that the threshold is chosen suitably. In Section 3.1, we obtain a fluid model of the system based on a differential equation arising from a fluid limit. This fluid model is used in Section 3.2 to prove that there exist thresholds such that the solutions to the differential equation converge over time to the even distribution of the load q^*. In particular, we prove that $\ell = \lfloor \lambda \rfloor$ has this property for all $\lambda \geq 0$, and that it is the unique threshold with this property unless $\lambda \in \mathbb{N}$; in the latter special case, $\ell = \lambda - 1$ is also fluid-optimal. In Section 3.3, we establish that $\ell = \lfloor \lambda \rfloor$ is optimal on the diffusion scale for all $\lambda \geq 0$.

3.1. Fluid Limit

Next we state a functional strong law of large numbers, also called the *fluid limit*, for the occupancy processes $q_n = \{q_n(i) : i \geq 0\}$. These processes take values in

$$Q = \left\{ q \in [0,1]^{\mathbb{N}} : q(i+1) \leq q(i) \leq q(0) = 1 \text{ for all } i \geq 1 \right\} \subset \mathbb{R}^{\mathbb{N}},$$

and their sample paths can be constructed on a common probability space for all n, as described in Section A.1 of the online supplement. The space of sample paths over a finite interval of time $[0, T]$ may be endowed with the metric ρ_u^∞ defined in Section A.2 of the online supplement, which corresponds to uniform convergence in the product topology of $\mathbb{R}^{\mathbb{N}}$. The next result holds for any finite time horizon T, any

threshold $\ell \in \mathbb{N}$, and any random limiting initial condition $q(0)$; a proof is provided in Section A.3 of the online supplement. Informally, this result establishes that the processes q_n approach solutions of a certain system of differential equations as n grows large.

Theorem 1. *Suppose that $q_n(0) \to q(0)$ in the product topology with probability one, and let $q_n|_{[0,T]}$ denote the restriction of q_n to $[0,T]$. Then $\{q_n|_{[0,T]} : n \geq 1\}$ is almost surely relatively compact with respect to the metric ρ_u^∞. Furthermore, the limit of each convergent subsequence is a function $q : [0,T] \to Q$ with Lipschitz coordinates and such that*

$$\dot{q}(i) = \lambda p_i(q,\ell) - i\big[q(i) - q(i+1)\big] \quad \text{for all} \quad i \geq 1 \qquad (2)$$

almost everywhere on $[0,T]$, with respect to the Lebesgue measure. The functions p_i in the aforementioned differential equations are defined as follows:

a. *If $q(\ell) < 1$, then*

$$p_i(q,\ell) = \begin{cases} \dfrac{q(i-1) - q(i)}{1 - q(\ell)} & \text{if} \quad 1 \leq i \leq \ell, \\ 0 & \text{if} \quad i \geq h. \end{cases}$$

b. *If $q(\ell) = 1$ and $q(h) < 1$, then*

$$p_i(q,\ell) = \begin{cases} \dfrac{\ell}{\lambda}\big[1 - q(h)\big] & \text{if} \quad i = \ell, \\ 1 - \dfrac{\ell}{\lambda}\big[1 - q(h)\big] & \text{if} \quad i = h, \\ 0 & \text{if} \quad i \neq \ell, h. \end{cases}$$

c. *If $q(h) = 1$, then*

$$p_i(q,\ell) = \begin{cases} \dfrac{h}{\lambda}\big[1 - q(h+1)\big] & \text{if} \quad i = h, \\ \Big[1 - \dfrac{h}{\lambda}\big(1 - q(h+1)\big)\Big]\big[q(i-1) - q(i)\big] & \text{if} \quad i \geq h+1, \\ 0 & \text{if} \quad 1 \leq i \leq \ell. \end{cases}$$

The fluid dynamics of Equation (2) have a simple interpretation: the derivative of $q(i)$ is the rate at which new tasks arrive to server pools with exactly $i-1$ tasks minus the rate at which tasks leave from server pools with precisely i tasks. The term $i[q(i) - q(i+1)]$ corresponds to the cumulative departure rate from server pools with exactly i tasks. Indeed, this quantity equals the total number of tasks in server pools with precisely i tasks, and each of these tasks has unit departure rate. The term $p_i(q,\ell)$ may be interpreted as the probability that a new task is assigned to a server pool with exactly $i-1$ tasks in fluid state q with threshold ℓ in force, and thus $\lambda p_i(q,\ell)$ corresponds to the arrival rate of tasks to server pools with $i-1$ tasks. More specifically, the expressions listed in Theorem 1 correspond to the following situations, respectively:

a. If $q(\ell) < 1$, then new tasks are sent to server pools with strictly less than ℓ tasks, chosen uniformly at random. So $p_i(q,\ell) = 0$ if $i \geq h$ and $p_i(q,\ell)$ is the fraction of pools with exactly $i-1$ tasks divided by the fraction of pools with at most $\ell - 1$ tasks if $1 \leq i \leq \ell$.

b. If $q(\ell) = 1$, then the arrival rate to server pools with precisely $\ell - 1$ tasks must equal the departure rate from server pools with exactly ℓ tasks, which gives $p_\ell(q,\ell)$. If $q(h) < 1$ and all server pools have at least ℓ tasks, then incoming tasks are sent to server pools with exactly ℓ tasks, so $p_h(q,\ell) = 1 - p_\ell(q,\ell)$ and $p_i(q,\ell) = 0$ for all $i \neq \ell, h$.

c. If $q(h) = 1$, then the right-hand side of (2) equals zero for $i = h$, which yields $p_h(q,\ell)$. The incoming tasks that are not sent to server pools with exactly ℓ tasks are sent to server pools with h or more tasks, and this happens with probability $1 - p_h(q,\ell)$. Among these server pools, a server pool with exactly $i > h$ tasks is chosen with probability $q(i-1) - q(i)$, which corresponds to uniform random routing.

The quantity $p_i(q,\ell)$ may lie outside $[0,1]$ for some i, q, and ℓ. However, if $q : [0,T] \to Q$ is the limit of a convergent subsequence as in Theorem 1, then $p_i(q(t),\ell) \in [0,1]$ for all $i \geq 1$ and all $t \in [0,T]$ outside a set of zero Lebesgue measure.

The proof of Theorem 1 uses a methodology developed in Bramson (1998) to prove the almost-sure relative compactness of sample paths and to establish that the limit of each convergent subsequence has Lipschitz coordinates. These limits are then characterized by a careful analysis in neighborhoods of the points where the derivatives of all coordinates exist, using the stochastic dynamics of the system. The differential Equation (2) results from this analysis; details are provided in Section A.3 of the online supplement.

3.2. Fluid-Optimal Thresholds

A threshold ℓ is said to be *fluid-optimal* if all solutions $q : [0,\infty) \to Q$ of (2) satisfy

$$\lim_{t\to\infty} q(t,i) = q^*(i) \quad \text{for all} \quad i \geq 0;$$

recall that q^* was defined in (1) and optimizes the overall quality of service.

Remark 1. Theorem 1 implies that solutions to (2), defined on $[0,\infty)$, exist for any threshold and initial condition; we do not claim, however, that these solutions are unique.

To identify the fluid-optimal thresholds, we fix $\ell \in \mathbb{N}$ and a solution $q : [0,\infty) \to Q$ of (2). We introduce the next functions to analyze how q evolves over time.

Definition 1. Define $u, v_j : [0,\infty) \to \mathbb{R}$ by $u(t) = \sum_{i=1}^\infty q(t,i)$ and $v_j(t) = \sum_{i=j}^\infty q(t,i)$ for all $j \geq 1$. These are the total mass function and tail mass functions, respectively.

The total mass function u corresponds to the total number of tasks normalized by the number of server pools. The tail mass function v_j has a similar interpretation if we visualize tasks as in Figure 1. Namely, it

corresponds to the total number of tasks located in column j of the diagram or further to the right, also normalized by the number of server pools.

By taking the sum over $i \geq 1$ on both sides of (2), we obtain $\dot{u}(t) = \lambda - u(t)$ for all $t \geq 0$. This differential equation corresponds to the fluid limit of the total number of tasks in an infinite-server system, which makes obvious sense, reflecting a prelimit dynamics property. Thus, $u(t) = \lambda + [u(0) - \lambda]e^{-t}$ for all $t \geq 0$, and, in particular, $u(t) \to \lambda$ as $t \to \infty$. We will use this fact to establish asymptotic upper bounds for $v_h(t)$ and $v_{h+1}(t)$ when $h \geq \lambda$.

Proposition 1. *Let x^+ denote $\max\{x, 0\}$. If $h > \lambda$, then there exists $t_0 \geq 0$ such that*

$$v_h(t) \leq (\lambda - \ell)^+ + e^{-(t-t_0)}\left[v_h(t_0) - (\lambda - \ell)^+\right], \tag{3a}$$

$$v_{h+1}(t) \leq e^{-(t-t_0)} v_{h+1}(t_0), \tag{3b}$$

for all $t \geq t_0$. The last inequality also holds if $h \geq \lambda$, with $t_0 = 0$.

Proof. Summing both sides of (2) over $i \geq h+1$, we obtain

$$\begin{aligned} \dot{v}_{h+1}(t) &= \left[\lambda - h(1 - q(t, h+1))\right]\mathbb{1}_{\{q(t,h)=1\}} - hq(t, h+1) \\ -v_{h+1}(t) &\leq \left[\lambda - h(1 - q(t,h+1))\right]^+ - hq(t, h+1) \\ -v_{h+1}(t) &\leq (\lambda - h)^+ - v_{h+1}(t). \end{aligned} \tag{4}$$

If $h \geq \lambda$, then $(\lambda - h)^+ = 0$ and we get (3b) with $t_0 = 0$. If $h > \lambda$, then $q(t,h) \leq u(t)/h$ and $u(t) \to \lambda$ as $t \to \infty$ imply that there exists $t_0 \geq 0$ such that $q(t,h) < 1$ for all $t \geq t_0$. Similar to (4), we conclude that

$$\begin{aligned} \dot{v}_h(t) &\leq \left[\lambda - \ell(1 - q(t,h))\right]^+ - \ell q(t,h) - v_h(t) \\ &\leq (\lambda - \ell)^+ - v_h(t) \\ &\text{for all} \quad t \geq t_0. \end{aligned}$$

This implies (3a), and clearly (3b) also holds. □

A consequence of this proposition is that the fraction of server pools with more than h tasks vanishes as $t \to \infty$ when $h \geq \lambda$. We will use Proposition 1 to prove that, for some suitable values of the threshold, $q(t,i) \to q^*(i)$ as $t \to \infty$ for all i.

Theorem 2. *We have that $\ell = \lfloor \lambda \rfloor$ is fluid-optimal for all λ and that $\ell = \lambda - 1$ is fluid-optimal if $\lambda \in \mathbb{N}$.*

Proof. Suppose that $\ell = \lfloor \lambda \rfloor$. Proposition 1 implies that $v_{h+1}(t) \to 0$ as $t \to \infty$, and hence

$$\begin{aligned} \liminf_{t\to\infty} \left[q(t,h) + \sum_{i=1}^{\ell} q(t,i)\right] &= \lim_{t\to\infty} \left[u(t) - v_{h+1}(t)\right] \\ &= \lim_{t\to\infty} u(t) = \lambda. \end{aligned}$$

It follows from Proposition 1 that the limit superior of $q(t, h)$ is upper-bounded by $\lambda - \lfloor \lambda \rfloor$, and evidently $\sum_{i=1}^{\ell} q(t,i) \leq \ell = \lfloor \lambda \rfloor$. Thus, $q(t,h) \to \lambda - \lfloor \lambda \rfloor$ and $\sum_{i=1}^{\ell} q(t,i) \to \lfloor \lambda \rfloor$ as $t \to \infty$. This implies that $q(t,i) \to q^*(i)$ as $t \to \infty$ for all i. The proof is similar when $\lambda \in \mathbb{N}$ and the threshold is $\ell = \lambda - 1$. □

The following two examples show that the thresholds mentioned in Theorem 2 are the only thresholds for which all solutions of (2) converge to q^*. In both cases, we exhibit equilibrium solutions of (2) that are different from q^*.

Example 1 (Large Threshold). Suppose that $\ell > \lfloor \lambda \rfloor$, and let $\theta \in (0,1)$ be a solution of

$$\frac{\lambda}{x} = \sum_{i=1}^{\ell} \frac{\ell!}{(\ell - i)!}\left(\frac{1-x}{\lambda}\right)^{i-1}. \tag{5}$$

Such a solution always exists, because the right-hand side has a finite limit as $x \to 0^+$, and it converges to $\ell > \lambda$ as $x \to 1^-$. Let us define $q(0) \in Q$ such that $q(0,i) = 0$ for all $i > \ell$ and

$$q(0, \ell - i) - q(0, \ell - (i-1)) = \frac{\ell!}{(\ell - i)!}\left(\frac{1-\theta}{\lambda}\right)^i \theta$$

for all $0 \leq i \leq \ell$. In order to see that $q(0)$ indeed lies in Q, observe that $q(0,\ell) = \theta$ and that

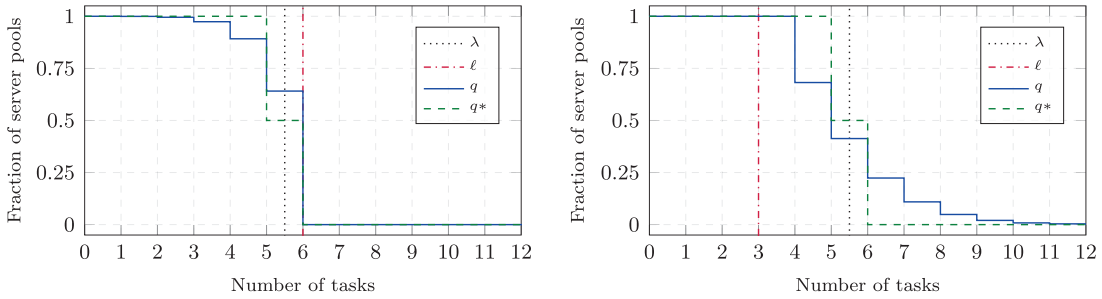
$$\begin{aligned} 1 = \theta + \sum_{i=1}^{\ell} \frac{\ell!}{(\ell - i)!}\left(\frac{1-\theta}{\lambda}\right)^i \theta &= \sum_{i=0}^{\infty} \left[q(0,i) - q(0,i+1)\right] \\ &= q(0,0), \end{aligned}$$

where the first equality follows from (5). Also, if we let $\pi(j) = q(0,j) - q(0,j+1)$ for all $0 \leq j \leq \ell$, then π is the stationary distribution of an Erlang-B system with ℓ servers and offered traffic $\lambda/(1-\theta)$. Moreover, $\theta = q(0,\ell)$ is the blocking probability of this system. Loosely speaking, each server pool behaves as a blocking system with ℓ servers and an offered traffic that is larger than λ, because tasks that find a full server pool are not discarded but sent to a server pool with idle servers.

It is possible to verify that our construction of the initial condition $q(0)$ is a fixed point of (2), so the constant function $q : [0, \infty) \to Q$ such that $q(t) = q(0)$ for all $t \geq 0$ solves (2). In addition, note that $q(0)$ corresponds to a suboptimal distribution of the load where the fraction of server pools with exactly i tasks is positive for all $0 \leq i \leq \ell$; see Figure 2.

In the previous example, the maximum number of tasks across server pools is $\ell > \lfloor \lambda \rfloor$. If the goal is to avoid concentrations of tasks, then this situation is nearly optimal when ℓ is close to $\lfloor \lambda \rfloor$; note that the even distribution of the load q^* achieved when the threshold $\ell = \lfloor \lambda \rfloor$ is fluid-optimal involves server pools with $\lfloor \lambda \rfloor + 1$ tasks. The most problematic situations arise, instead, when the threshold is lower than the optimal value. Intuitively, in these cases all server pools have more than ℓ tasks most of the time, because the threshold is smaller than the load. As a result, the system has to resort to random routing

Figure 2. (Color online) Equilibrium Solutions Computed in Examples 1 and 2 for $\lambda = 5.5$



Note. The left plot corresponds to the case of a large threshold, whereas the right plot corresponds to the case of a small threshold.

very often, which is known to be highly inefficient. In the large-scale limit, this translates into the number of tasks across server pools being unbounded.

Example 2 (Small Threshold). Suppose that $\ell < \lfloor \lambda \rfloor$ and $\lambda \notin \mathbb{N}$ or, alternatively, that $\ell < \lambda - 1$ and $\lambda \in \mathbb{N}$. In addition, let $\theta \in (0,1)$ solve

$$\sum_{i=h}^{\infty} \frac{h!}{i!}\left[\lambda - h(1-x)\right]^{i-h}(1-x) = 1. \tag{6}$$

A solution always exists, because the left-hand side is strictly larger than one for $x = 0$ and equal to zero for $x = 1$. We define $q(0) \in Q$ such that $q(0,i) = 1$ for all $1 \le i \le h$ and

$$q(0,i) - q(0,i+1) = \frac{h!}{i!}\left[\lambda - h(1-\theta)\right]^{i-h}(1-\theta) \text{ for all } i \ge h.$$

By setting $i = h$ in the last equation, we observe that $1 - \theta$ is the fraction of server pools with at most h tasks. Furthermore, using (6), we conclude that $\pi(j) = q(0,j) - q(0,j+1)$ for all $j \ge h$ is the stationary distribution of the birth-death process with death rate j at state j and birth rate $\lambda - h(1-\theta)$ at each state. Loosely speaking, tasks are sent to a server pool chosen uniformly at random when all server pools have at least h tasks. The fraction of server pools with at least h tasks operates at one, and server pools with exactly ℓ tasks receive new tasks at rate $\lambda p_h(q(0),\ell) = h(1-\theta) < \lambda$. The remaining tasks are sent to a server pool chosen uniformly at random, so server pools with h or more tasks receive new tasks at rate $\lambda - h(1-\theta)$.

As in Example 1, the initial condition $q(0)$ is a fixed point of (2), and so the constant function $q : [0,\infty) \to Q$ such that $q(t) = q(0)$ for all $t \ge 0$ solves (2). Also, $q(0)$ is an occupancy state for which the fraction of server pools with exactly i tasks is positive for all $i \ge h$; an occupancy state of this kind is depicted in Figure 2.

The next corollary is a consequence of Theorem 2 and the two examples.

Corollary 1. *If $\lambda \notin \mathbb{N}$, then $\lfloor \lambda \rfloor$ is the unique fluid-optimal threshold. In the special case where $\lambda \in \mathbb{N}$, there exist exactly two fluid-optimal thresholds: $\lambda - 1$ and λ.*

3.3. Diffusion-Scale Optimality

We have already proved that our policy is fluid-optimal when $\ell = \lfloor \lambda \rfloor$, and we establish here that our policy is optimal on the diffusion scale as well for this value of the threshold. This is done by proving that our policy has the same behavior as JSQ on the diffusion scale; indeed, the optimality properties of JSQ established in Menich and Serfozo (1991) and Sparaggis et al. (1993) are stronger than diffusion-scale optimality and, in particular, imply that JSQ is optimal on the diffusion scale. Specifically, suppose that $\lambda \notin \mathbb{N}$, and let

$$\bar{Y}_n = \sum_{i=1}^{\lfloor \lambda \rfloor} \frac{n - Q_n(i)}{\log n}, \quad \bar{Z}_n = \frac{Q_n(\lceil \lambda \rceil) - (\lambda - \lfloor \lambda \rfloor)n}{\sqrt{n}}, \qquad \bar{Q}_n(i) = \frac{Q_n(i)}{\sqrt{n}} \quad \text{for all} \quad i > \lceil \lambda \rceil. \tag{7}$$

These random variables have the same asymptotic behavior both for JSQ and our threshold-based policy, as stated in the following theorem.

Theorem 3. *Suppose that $\lambda \notin \mathbb{N}$, $\ell = \lfloor \lambda \rfloor$, and that $\bar{Y}_n(0)$, $\bar{Z}_n(0)$, and $\bar{Q}_n(0,i)$, for all $i > \lceil \lambda \rceil$, have a limit in distribution as $n \to \infty$; denote these limits by $\bar{Y}(0)$, $\bar{Z}(0)$, and $\bar{Q}(0,i)$, respectively.*

a. *The sequence $\{\bar{Y}_n : n \ge 1\}$ is stochastically bounded in the Skorokhod space $D_{\mathbb{R}}[0,\infty)$.*

b. *$\bar{Z}_n$ converges weakly in $D_{\mathbb{R}}[0,\infty)$ as $n \to \infty$ to an Ornstein-Uhlenbeck process Z, which satisfies the stochastic differential equation $dZ = -Z dt + \sqrt{2\lambda} dW$; here W denotes a standard Wiener process.*

c. *Provided that $\bar{Q}_n(0,i)$ converges in probability to zero, $\bar{Q}_n(i)$ converges weakly in $D_{\mathbb{R}}[0,\infty)$ to the identically zero process; this holds for all $i > \lceil \lambda \rceil$.*

The proof of Theorem 3 is carried out in Section B.1 of the online supplement through a stochastic coupling between a system that uses our policy and a

system that uses the JSQ policy; the diffusion limit of JSQ, which coincides with Theorem 3, was derived in Mukherjee et al. (2020). Loosely speaking, Theorem 3 implies that, for large enough n, and after sufficient time, the number of server pools with $\lceil\lambda\rceil$ tasks is $(\lambda - \lfloor\lambda\rfloor)n + O(\sqrt{n})$ and the number of server pools with fewer than $\lfloor\lambda\rfloor$ tasks is $O(\log n)$. Also, if the system starts with no server pools with more than $\lceil\lambda\rceil$ tasks, then it will remain so.

The diffusion-scale optimality of the threshold $\ell = \lfloor\lambda\rfloor$ can also be established in the special case $\lambda \in \mathbb{N}$, and this is done in Appendix B of the online supplement.

4. Learning the Optimal Threshold

In this section, we propose a control rule for adjusting a dynamic threshold over time, to learn the optimal threshold value when λ is unknown. This rule is explained in Section 4.1 and then analyzed through a fluid-limit approach in the following sections. Specifically, a fluid model for systems with dynamic thresholds is introduced in Section 4.2 and is then justified in Section 4.3 through a fluid limit. The evolution of the dynamic threshold over time is analyzed through the fluid model in Section 4.4, where we establish that the threshold always reaches an equilibrium. In Section 4.5, we explain how to tune our learning rule so that the equilibrium threshold is always nearly optimal, and we prove that this tuning yields in fact an optimal equilibrium threshold in most situations. In addition, the time required for the threshold to settle is analyzed in Section 4.6.

4.1. Learning Rule

In Section 3, we showed that our threshold policy is fluid and diffusion-optimal, provided that $\ell = \lfloor\lambda\rfloor$. However, these optimality properties critically rely on the threshold being strictly equal to $\lfloor\lambda\rfloor$, as was shown by Examples 1 and 2. Furthermore, in actual system deployments, discrepancies between an a priori chosen threshold and the optimal value $\lfloor\lambda\rfloor$ may occur due to the following two reasons:

1. In general, it is difficult to estimate λ in advance, and a slightly inaccurate estimate may result in a wrong choice of the threshold. The worst performance repercussions occur if λ is underestimated and a low threshold is chosen, as explained before Example 2.

2. Even if the threshold equals $\lfloor\lambda\rfloor$ initially, λ could change due to fluctuations in the overall demand for service. These fluctuations could result in a mismatch between ℓ and $\lfloor\lambda\rfloor$, with the corresponding adverse consequences in terms of performance.

Remark 2. We have adopted the common assumption of unit-mean service times, which amounts to a convenient choice of time unit. In view of this, it is worth observing that the optimal threshold is determined by the offered load, rather than the arrival rate of tasks. Specifically, if service times had mean $1/\mu$, then the optimal threshold would be $\ell = \lfloor\rho\rfloor$, with $\rho = \lambda/\mu$ the offered load. In particular, it is the offered load that has to be estimated rather than the arrival rate of tasks, which exacerbates the issues mentioned earlier. Although we assume unit-mean service times, our results easily generalize to a generic service rate μ without changing the control rule that we describe next, which is designed to track the offered load rather than the arrival rate of tasks.

To achieve optimality, it is necessary to actively learn the optimal threshold. To this purpose, we propose a control rule for adjusting the threshold in an online fashion. To explain this rule, let us denote the threshold in a system with n server pools by $\ell_n(t)$, which is now *time-dependent*; as before, we also introduce the notation $h_n(t) = \ell_n(t) + 1$ for convenience. Our control rule depends on a parameter $\alpha \in (0,1)$, and it adjusts the threshold only at arrival epochs, right after dispatching the new task. If an arrival occurs at time τ, then the threshold is adjusted as follows:

- The threshold is increased by one if the number of server pools with at least h_n tasks, measured right before time τ, is greater than or equal to $n - 1$.
- The threshold is decreased by one if the fraction of server pools with at least ℓ_n tasks, measured right before time τ, is smaller than or equal to α.
- Otherwise, the threshold remains unchanged.

This control rule only relies on knowledge of the tokens that are maintained by the dispatcher when the implementation of Section 2.2 is adopted. Specifically, the threshold is increased if and only if the number of yellow tokens was smaller than or equal to one prior to the arrival, and the threshold is decreased if and only if the number of green tokens was larger than or equal to $(1 - \alpha)n$ right before the arrival occurred.

Suppose that λ is unknown, either because it was not possible to estimate the offered load in advance or because it recently changed. As tasks arrive to the system, the control rule that we have just described adjusts the threshold in steps of one unit. At this point, the most relevant question is whether it is possible to choose α so that these updates eventually cease, with the threshold settling at the optimal value. Another important question is how long it takes for the threshold to settle. Indeed, after each update, the new threshold must be communicated to all server pools and the information stored at the dispatcher must be updated. Fast convergence of the threshold is thus desired.

4.2. Fluid Systems

In this subsection, we introduce the notion of a *fluid system*, which will help us model the behavior of large-scale systems that use our adaptive threshold policy through differential equations.

Definition 2. Consider sequences $\{\tau_j : 0 \le j < \eta\}$ and $\{\ell_j \in \mathbb{N} : 0 \le j < \eta\}$ of increasing times and thresholds, respectively. Suppose that $\tau_0 = 0$, and let $\tau_\eta = \infty$ if $\eta < \infty$ and $\tau_\eta = \lim_{j\to\infty} \tau_j$ otherwise. We define $\ell : [0, \tau_\eta) \to \mathbb{N}$ such that $\ell(t) = \ell_j$ for all $t \in [\tau_j, \tau_{j+1})$ and all $0 \le j < \eta$, and given $q : [0, \tau_\eta) \to Q$, we say that $s = (q, \ell)$ is a *fluid system* if the following hold:

a. $q(t, \ell(t)) \ge \alpha$ for all $t \in [0, \tau_\eta)$ and $q(h) < 1$ almost everywhere on $[0, \tau_\eta)$ with respect to the Lebesgue measure; as usual, we define $h_j = \ell_j + 1$ and $h(t) = \ell(t) + 1$.

b. Either $q(\tau_{j+1}, \ell_j) = \alpha$ or $q(\tau_{j+1}, h_j) = 1$ for all $0 \le j < \eta - 1$. Moreover,

$$\ell_{j+1} = \ell_j - 1 \quad \text{if} \quad q(\tau_{j+1}, \ell_j) = \alpha,$$
$$\ell_{j+1} = \ell_j + 1 \quad \text{if} \quad q(\tau_{j+1}, h_j) = 1.$$

c. The coordinate functions $q(i)$ are absolutely continuous, and

$$\dot{q}(i) = \lambda p_i(q, \ell_j) - i\big[q(i) - q(i+1)\big] \quad \text{for all} \quad i \ge 1 \tag{8}$$

almost everywhere on $[\tau_j, \tau_{j+1})$ for all $0 \le j < \eta$.

A fluid system consists of a function q, which represents the system occupancy, and a piecewise constant function ℓ, which represents the time-dependent threshold; the times τ_j correspond to threshold updates. Between τ_j and τ_{j+1}, the threshold is constant, equal to ℓ_j, and the system behaves according to the differential equation of Theorem 1. In addition, the threshold is updated according to the control rule explained earlier: informally, the threshold increases when $q(h)$ reaches one, and it decreases when $q(\ell)$ drops below α.

In Definition 2, the possibly finite τ_η accounts for the possibility of infinitely many updates in finite time; we will prove, however, that this in fact cannot happen. To this end, we resort to the total mass function $u : [0, \tau_\eta) \to \mathbb{R}$ of Definition 1. As in Section 3.2,

$$u(t) = \lambda + [u(0) - \lambda]e^{-t} \quad \text{for all} \quad t \in [0, \tau_\eta). \tag{9}$$

Indeed, for each $0 \le j < \eta$, we see that $\dot{u}(t) = \lambda - u(t)$ for all $t \in [\tau_j, \tau_{j+1})$ by taking the sum over $i \ge 1$ on both sides of (8). The following proposition establishes that the threshold of a fluid system cannot change infinitely many times in finite time.

Proposition 2. *All fluid systems have* $\tau_\eta = \infty$.

Proof. Consider a fluid system with $\eta = \infty$; otherwise, the claim holds by Definition 2. It follows from (9) that $u(t)$ is upper-bounded by some constant $M \ge 0$. Since the sequence $q(t) \in Q$ is nonincreasing for each time t, we have $\alpha \le q(t, \ell_j) \le u(t)/\ell_j \le M/\ell_j$ for all $t \in [\tau_j, \tau_{j+1})$ and all $j \ge 0$. Therefore, $\{\ell_j : j \ge 0\}$ is a bounded set, and, as a result, the set $\mathcal{L} = \{\ell \in \mathbb{N} : \ell_j = \ell \text{ for infinitely many } j\}$ is nonempty and bounded. We define $m = \max \mathcal{L}$, and we observe that there exists $j_0 \ge 0$ such that $\ell_j \le m$ for all $j \ge j_0$.

Fix any index $j > j_0$ such that $\ell_j = m$. By (8), we know that

$$\dot{q}(m) = \lambda p_m(q, \ell_j) - m\big[q(m) - q(m+1)\big] \ge -m$$

almost everywhere on $[\tau_j, \tau_{j+1})$. The thresholds ℓ_{j-1} and ℓ_{j+1} are equal to $m - 1$ by definition of j_0. In particular, $q(\tau_j, m) = 1$ and $q(\tau_{j+1}, m) = \alpha$, which implies that

$$\alpha = q(\tau_{j+1}, m) \ge q(\tau_j, m) - m(\tau_{j+1} - \tau_j) = 1 - m(\tau_{j+1} - \tau_j).$$

By the definition of m, there exist infinitely many indexes $j > j_0$ such that $\ell_j = m$. For these indexes, we have proved that $\tau_{j+1} - \tau_j \ge (1 - \alpha)/m$, and therefore $\tau_j \to \infty$ as $j \to \infty$. □

4.3. Fluid Limit

Next, we provide a fluid limit that justifies the use of fluid systems as an asymptotic approximation to the behavior of the discrete system (q_n, ℓ_n) as n grows large. Before stating this fluid limit, we must introduce a few technical definitions. Namely, for each finite time horizon T, consider the space $D_{\mathbb{R}}[0, T]$ of all real càdlàg functions on $[0, T]$, and endow it with the metric ρ_s defined in (Billingsley 2013, section 12), which is complete and corresponds to the Skorokhod J_1-topology. In addition, recall the metric ρ_u^∞ mentioned in Section 3.1 and defined rigorously in Section A.2 of the online supplement; this metric is defined on the Skorokhod space $D_{\mathbb{R}^{\mathbb{N}}}[0, T]$ of all càdlàg functions with values in $\mathbb{R}^{\mathbb{N}}$.

For each discrete system $s_n = (q_n, \ell_n)$, the restriction $s_n|_{[0,T]}$ is a random function lying in the space $S = D_{\mathbb{R}^{\mathbb{N}}}[0, T] \times D_{\mathbb{R}}[0, T]$, which we endow with the following metric:

$$\varrho((p, x), (q, y)) = \max\{\rho_u^\infty(p, q), \rho_s(x, y)\}.$$

Observe that a sequence $\{(p_n, x_n) \in S : n \ge 1\}$ converges with respect to ϱ if and only if p_n converges uniformly over $[0, T]$, with respect to the product topology of $\mathbb{R}^{\mathbb{N}}$, and x_n converges in the Skorokhod J_1-topology.

As in Section 3.1, it is possible to construct the sample paths of s_n on a common probability space for all n. We adopt such a construction to state the next result, which holds for any pair of random elements $q(0)$ and $\ell(0)$ taking values in Q and $\mathbb{N}$, respectively.

Theorem 4. *Suppose that* $q_n(0) \to q(0)$ *in the product topology and* $\ell_n(0) \to \ell(0)$ *almost surely as* $n \to \infty$. *Then* $\{s_n|_{[0,T]} : n \ge 1\}$ *is almost surely relatively compact with respect to* ϱ *and the limit* $s \in S$ *of each convergent subsequence can be extended to a fluid system.*

The almost-sure relative compactness of $\{q_n|_{[0,T]} : n \ge 1\}$ with respect to ρ_u^∞ can be obtained using the methodology of Bramson (1998), as indicated in Section 3.1. Given a sample path such that the latter relative compactness holds, the challenge is to

demonstrate that, for each convergent subsequence $\{q_{n_k}|_{[0,T]} : k \geq 1\}$, the associated thresholds converge with respect to ρ_s, and to characterize the limits of $q_{n_k}|_{[0,T]}$ and $\ell_{n_k}|_{[0,T]}$ jointly. This is done inductively, by approaching s_n by systems where only finitely many threshold updates occur, starting with systems where the threshold is constant over time as in Section 3.1. The proof of this result is provided in Section A.4 of the online supplement.

4.4. Convergence of the Threshold

In the statement of Theorem 4, the limit s of a convergent subsequence is the restriction to the interval $[0,T]$ of a fluid system. Next, we will show that the time-dependent threshold of any fluid system eventually settles at an equilibrium value. For this purpose, we fix a fluid system $s = (q, \ell)$, and we consider the associated total mass and tail mass functions, as in Definition 1. The next result provides upper bounds for the tail mass functions.

Proposition 3. *Suppose that there exist $m \geq 0$ and $0 \leq a < b$ such that $\ell(t) \leq m$ for all $t \in (a,b)$. Then the following inequalities hold for all $t \in [a,b]$:*

$$v_{m+1}(t) \leq (\lambda - m)^+ + \mathrm{e}^{-(t-a)}\big[v_{m+1}(a) - (\lambda - m)^+\big], \quad (10a)$$

$$v_{m+2}(t) \leq \mathrm{e}^{-(t-a)} v_{m+2}(a). \quad (10b)$$

If, in addition, $q(t,m) < 1$ for all $t \in (a,b)$, then

$$v_{m+1}(t) \leq \mathrm{e}^{-(t-a)} v_{m+1}(a) \quad \text{for all} \quad t \in [a,b]. \quad (11)$$

Proof. Recall from Definition 2 that $q(h) < 1$ almost everywhere. For each $t \in (a,b)$, we have $h(t) \leq m+1$, so $\lambda p_{m+1}(s) \leq \big[\lambda - m(1 - q(m+1))\big]^+$ and $p_i(s) = 0$ for all $i \geq m+2$ almost everywhere on (a, b). The proof of (10) proceeds as in Proposition 1, and (11) follows similarly, by noting that $q(m) < 1$ implies that $p_i(s) = 0$ for all $i \geq m+1$. □

We now prove that the threshold settles; the next result holds for all $\lambda \geq 0$.

Theorem 5. There exist $t_{\mathrm{eq}} \geq 0$ and $\ell_{\mathrm{eq}} \in \mathbb{N}$ such that $\ell(t) = \ell_{\mathrm{eq}}$ for all $t \geq t_{\mathrm{eq}}$.

Proof. By (9), there exists $t_0 \geq 0$ such that $u(t) < \lfloor \lambda \rfloor + 1$ for all $t \geq t_0$. Hence,

$$\tau_j \geq t_0 \quad \text{and} \quad \ell_j \geq \lfloor \lambda \rfloor \quad \text{imply}$$

$$q(t, h_j) \leq \frac{u(t)}{h_j} < 1 \quad \text{for all} \quad t \in [\tau_j, \tau_{j+1}). \quad (12)$$

Hence, one of the next situations occurs: the next threshold update happens at $\tau_{j+1} < \infty$ and it corresponds to a threshold decrease, or no further updates occur and $\tau_{j+1} = \infty$.

We now consider two alternative scenarios. First, suppose that $\ell(t) \geq \lfloor \lambda \rfloor$ for all $t \geq t_0$. The previous observation implies that ℓ is nonincreasing, integer-valued, and lower-bounded along the interval $[t_0, \infty)$. Therefore, $\ell(t)$ settles at some $\ell_{\mathrm{eq}} \geq \lfloor \lambda \rfloor$. Alternatively, assume that there exists $t_1 \geq t_0$ such that $\ell(t_1) < \lfloor \lambda \rfloor$. If ℓ reaches $\lfloor \lambda \rfloor$ after t_1, then ℓ cannot increase any further by (12), so $\ell(t) \leq \lfloor \lambda \rfloor$ for all $t \geq t_1$.

In the latter case, Proposition 3 holds with $m = \lfloor \lambda \rfloor$, $a = t_1$, and $b = \infty$. In particular,

$$v_{\lfloor \lambda \rfloor + 1}(t) \leq \lambda - \lfloor \lambda \rfloor + \big[v_{\lfloor \lambda \rfloor + 1}(t_1) - (\lambda - \lfloor \lambda \rfloor)\big]\mathrm{e}^{-(t-t_1)}$$

for all $t \geq t_1$. The right-hand side converges to $\lambda - \lfloor \lambda \rfloor$ over time, and $u(t) \to \lambda$ over time by (9). As a result, there exists $t_2 \geq t_1$ such that $u(t) - v_{\lfloor \lambda \rfloor + 1}(t) > \lfloor \lambda \rfloor - (1 - \alpha)$ for all $t \geq t_2$.

Suppose that $\tau_j \geq t_2$. Then $q(t, \ell_j) \geq u(t) - (\lfloor \lambda \rfloor - 1) - v_{\lfloor \lambda \rfloor + 1}(t) > \alpha$ for all $t \in [\tau_j, \tau_{j+1})$, because $\ell_j \leq \lfloor \lambda \rfloor$. This implies that one of the following two situations occurs: the next threshold update happens at $\tau_{j+1} < \infty$ and it is a threshold increase, or no further updates occur and $\tau_{j+1} = \infty$.

Concluding, if $\ell(t_1) < \lfloor \lambda \rfloor$ for some $t_1 \geq t_0$, then there exists $t_2 \geq t_1$ such that the threshold is nondecreasing along the interval $[t_2, \infty)$ and is bounded above by $\lfloor \lambda \rfloor$. Thus, ℓ settles in this case as well, at some value $\ell_{\mathrm{eq}} \leq \lfloor \lambda \rfloor$. □

4.5. Tuning of the Learning Rule

Theorem 5 does not provide any information about ℓ_{eq} or t_{eq}. Particularly, we would like to know how these values depend on α, to set this parameter in a suitable manner. To shed some light on this matter, we first investigate the possible values of ℓ_{eq}. We provide lower and upper bounds, which are in turn used to obtain a criterion for setting α.

Proposition 4. *If $\lambda \notin \mathbb{N}$, then $\ell_{\mathrm{eq}} \geq \lfloor \lambda \rfloor$, and if $\lambda \in \mathbb{N}$, then $\ell_{\mathrm{eq}} \geq \lambda - 1$.*

Proof. Let us define $h_{\mathrm{eq}} = \ell_{\mathrm{eq}} + 1$ and suppose that $\lambda \notin \mathbb{N}$; the proof is similar if $\lambda \in \mathbb{N}$. Proposition 3, with $m = \ell_{\mathrm{eq}}$, $a = t_{\mathrm{eq}}$, and $b = \infty$, implies that $v_{h_{\mathrm{eq}}+1}(t) \leq v_{h_{\mathrm{eq}}+1}(t_{\mathrm{eq}})\mathrm{e}^{-(t-t_{\mathrm{eq}})}$ for all $t \geq t_{\mathrm{eq}}$. This in turn implies that $u(t) - v_{h_{\mathrm{eq}}+1}(t) \to \lambda$ over time. As a result, we have

$$h_{\mathrm{eq}} \geq \limsup_{t \to \infty} \sum_{i=1}^{h_{\mathrm{eq}}} q(t,i) = \lim_{t \to \infty} \big[u(t) - v_{h_{\mathrm{eq}}+1}(t)\big] = \lambda.$$

Since $\lambda \notin \mathbb{N}$, we conclude that $h_{\mathrm{eq}} \geq \lceil \lambda \rceil$, and thus $\ell_{\mathrm{eq}} \geq \lfloor \lambda \rfloor$. □

Proposition 5. *The equilibrium threshold satisfies $\ell_{\mathrm{eq}} \leq \lambda/\alpha$, both for $\lambda \notin \mathbb{N}$ and $\lambda \in \mathbb{N}$.*

Proof. Recall from the proof of Proposition 4 that $u(t) - v_{h_{\mathrm{eq}}+1}(t) \to \lambda$ as $t \to \infty$. Also, $q(t,i) \geq q(t, \ell_{\mathrm{eq}}) \geq \alpha$ for all $t \geq t_{\mathrm{eq}}$ and all $i \leq \ell_{\mathrm{eq}}$ by Definition 2.

Consequently,

$$0 \le \limsup_{t\to\infty} q(t, h_{\text{eq}}) = \limsup_{t\to\infty} \left[u(t) - v_{h_{\text{eq}}+1}(t) - \sum_{i=1}^{\ell_{\text{eq}}} q(t,i) \right] \le \lambda - \alpha \ell_{\text{eq}},$$

and this completes the proof. □

Assume that an upper bound of λ is known, say, $\lambda_{\max}$. We propose to set α such that

$$\alpha > \frac{\lambda_{\max}}{\lambda_{\max}+1}; \tag{13}$$

the assumption that $\lambda \le \lambda_{\max}$ is not strong in practice, especially since the aforementioned criterion can be used even if the upper bound $\lambda_{\max}$ is chosen conservatively.

The right-hand side of (13) is increasing in $\lambda_{\max}$, which implies that $\alpha > \lambda/(\lambda+1)$ for all offered loads $\lambda \le \lambda_{\max}$. Recall that $\ell_{\text{eq}} \le \lambda/\alpha < \lambda + 1$ by Proposition 5, and thus $\ell_{\text{eq}} \le \lceil \lambda \rceil$ for all λ; that is, the equilibrium threshold is at most the optimal threshold plus one. More specifically, $\lfloor \lambda \rfloor \le \ell_{\text{eq}} \le \lceil \lambda \rceil$ if $\lambda \notin \mathbb{N}$ and $\lambda - 1 \le \ell_{\text{eq}} \le \lambda$ in the special case $\lambda \in \mathbb{N}$.

We claim that this criterion guarantees nearly-optimal behavior on the fluid scale. To see why, observe that $\ell_{\text{eq}} \le \lceil \lambda \rceil$. Thus, Proposition 3, with $m = \lceil \lambda \rceil$, $a = t_{\text{eq}}$, and $b = \infty$, implies that $v_{\lfloor \lambda \rfloor + 2}(t) \le e^{-(t-t_{\text{eq}})} v_{\lfloor \lambda \rfloor + 2}(t_{\text{eq}})$ for all $t \ge t_{\text{eq}}$. Hence, after the threshold settles, the fraction of server pools with more than $\lfloor \lambda \rfloor + 1$ tasks decays at least exponentially fast to zero. Although the system may not attain the ideal distribution of the load defined in (1), the fraction of server pools with $\lceil \lambda \rceil$ tasks or more vanishes over time.

If (13) holds, then the threshold settles at a nearly-optimal value. However, in many situations, the threshold will in fact settle at the optimal value. This is established in the next corollary, which is a straightforward consequence of Propositions 4 and 5.

Corollary 2. *Suppose that*

$$\frac{\lambda}{\lfloor \lambda \rfloor + 1} < \alpha. \tag{14}$$

This condition implies that $\ell_{\text{eq}} = \lfloor \lambda \rfloor$ *if* $\lambda \notin \mathbb{N}$ *or* $\ell_{\text{eq}} \in \{\lambda - 1, \lambda\}$ *if* $\lambda \in \mathbb{N}$.

This corollary tells us that the optimality of the equilibrium threshold may be lost only when λ is close enough to an integer from below. For each α, it is possible to find values of λ that violate (13). However, the set of such λ decreases to the empty set as $\alpha \to 1$.

4.6. Convergence Time

Assuming that $\lambda \notin \mathbb{N}$ and that the optimality condition (14) holds, we now focus on the asymptotic time t_{eq} required by our learning rule to reach an equilibrium. In particular, the next proposition provides an upper bound $\bar{t}_{\text{eq}}$ for this time.

Proposition 6. *Suppose that* $\lambda \notin \mathbb{N}$ *and* (14) *holds. Then, for all*

$$t \ge \bar{t}_{\text{eq}} = \begin{cases} \log\left(\dfrac{\lambda}{\lambda - \lfloor \lambda \rfloor}\right) & \text{if} \quad u(0) \le \lambda, \\ \left[\log\left(\frac{u(0)-\lambda}{\alpha\lceil\lambda\rceil - \lambda}\right)\right]^+ + \log\left(\dfrac{\lambda}{\lambda - \lfloor \lambda \rfloor}\right) & \text{if} \quad u(0) > \lambda, \end{cases} \tag{15}$$

we have $\ell(t) = \lfloor \lambda \rfloor$ *and* $q(t, \lfloor \lambda \rfloor) = 1$. *In particular,* ℓ *settles at* $t_{eq} \le \bar{t}_{\text{eq}}$.

Proof. Similar to (12), we may write

$$q(t, \ell_j) \le \frac{u(t)}{\ell_j} = \frac{\lambda + [u(0) - \lambda] e^{-t}}{\ell_j} \quad \text{for all} \quad t \in [\tau_j, \tau_{j+1}) \quad \text{and} \quad 0 \le j < \eta;$$

where we used (9) for the last step. We now choose s_0 so that the right-hand side is strictly less than α if $\ell_j \ge \lceil \lambda \rceil$ and $t > s_0$. Specifically, using (14), we set

$$s_0 = 0 \quad \text{if} \quad u(0) \le \lambda,$$
$$s_0 = \left[\log\left(\frac{u(0) - \lambda}{\alpha\lceil\lambda\rceil - \lambda}\right)\right]^+ \quad \text{if} \quad u(0) > \lambda.$$

Note that $\ell(t) \le \lfloor \lambda \rfloor$ for all $t > s_0$ necessarily. To see why, recall that Definition 2 imposes $q(\ell) \ge \alpha$ at all times, and then note that $\ell(t) \ge \lceil \lambda \rceil$ and $t > s_0$ would imply $q(t, \ell(t)) < \alpha$. In the special case $\lfloor \lambda \rfloor = 0$, this observation implies that $\ell(t) = \lfloor \lambda \rfloor$ for all $t > s_0$. In particular, $\ell(t) = \lfloor \lambda \rfloor$ and $q(t, \lfloor \lambda \rfloor) = 1$ (trivially) for all $t \ge \bar{t}_{\text{eq}}$. Thus, we assume in what follows that $\lfloor \lambda \rfloor > 0$.

Let $w = u - v_{\lceil \lambda \rceil}$. Summing over $1 \le i \le \lfloor \lambda \rfloor$ on both sides of (8), we obtain

$$\dot{w}(t) = \sum_{i=1}^{\lfloor \lambda \rfloor} \dot{q}(t,i) \ge \lambda - \lfloor \lambda \rfloor > 0 \quad \text{whenever} \quad \ell(t) \le \lfloor \lambda \rfloor \quad \text{and} \quad q(t, \lfloor \lambda \rfloor) < 1,$$

which implies that w is nondecreasing after s_0. Indeed, $\ell(t) \le \lfloor \lambda \rfloor$ after s_0, so $\dot{w}$ is positive unless $q(\lfloor \lambda \rfloor) = 1$, which is equivalent to w attaining its maximum value $\lfloor \lambda \rfloor$.

Suppose that $q(t, \lfloor \lambda \rfloor) < 1$ for all $t \in [s_0, \bar{t}_{\text{eq}}]$. This implies that (11) holds with $m = \lfloor \lambda \rfloor$, $a = s_0$, and $b = \bar{t}_{\text{eq}}$. Using (9) as well, we arrive at the following contradiction:

$$w(\bar{t}_{\text{eq}}) = u(\bar{t}_{\text{eq}}) - v_{\lceil \lambda \rceil}(\bar{t}_{\text{eq}}) \ge \lambda + [u(s_0) - \lambda] e^{-(\bar{t}_{\text{eq}} - s_0)} - v_{\lceil \lambda \rceil}(s_0) e^{-(\bar{t}_{\text{eq}} - s_0)} \ge \lfloor \lambda \rfloor.$$

Thus, there exists $s_1 \in [s_0, \bar{t}_{\text{eq}}]$ such that $q(s_1, \lfloor \lambda \rfloor) = 1$. Since $w(s_1) = \lfloor \lambda \rfloor$ is at its maximum, $q(\lfloor \lambda \rfloor)$ cannot decrease. This implies that $\ell(t) = \lfloor \lambda \rfloor$ and $q(t, \lfloor \lambda \rfloor) = 1$ for all $t \ge \bar{t}_{\text{eq}}$. □

The expressions for $\bar{t}_{\text{eq}}$ provided in (15) consist of two terms: the first one upper-bounds the time until the threshold falls and remains below $\lfloor\lambda\rfloor$, and the second one accounts for the additional amount of time until the threshold reaches $\lfloor\lambda\rfloor$ and settles. The first term is zero when $u(0) \le \lambda$; in this case, ℓ can never exceed $\lfloor\lambda\rfloor$, since u is upper-bounded by λ; here the expression in (15) corresponds to the time needed by u to reach $\lfloor\lambda\rfloor$ when $u(0) = 0$. When $u(0) > \lambda$, the two terms are positive. The first one increases with the initial total mass, as one would expect, and, more interestingly, also depends on $\alpha\lceil\lambda\rceil - \lambda$. Loosely speaking, if the fractional part of λ is large, then it might take longer for $q(\lceil\lambda\rceil)$ to drop below α and trigger a threshold update from $\lceil\lambda\rceil$ to $\lfloor\lambda\rfloor$. The second term is the same as in the case where $u(0) \le \lambda$ and could possibly be reduced.

The next corollary uses the upper bound $\bar{t}_{\text{eq}}$ to summarize the asymptotic optimality properties of our policy when $\lambda \notin \mathbb{N}$ and (14) holds. Informally, this corollary states that the threshold settles at the optimal value $\lfloor\lambda\rfloor$ before $\bar{t}_{\text{eq}}$ in all sufficiently large systems and that the occupancy approaches q^* over time at least exponentially fast. The proof is a straightforward consequence of Theorem 4 and Propositions 3 and 6.

Corollary 3. *Suppose that $\lambda \notin \mathbb{N}$ and (14) holds. There exists a constant $c > 0$ such that*

$$\lim_{n\to\infty} \sup_{t\in[\bar{t}_{\text{eq}},T]} |\ell_n(t) - \lfloor\lambda\rfloor| = 0,$$

$$\lim_{n\to\infty} \sup_{t\in[\bar{t}_{\text{eq}},T]} |q_n(t,i) - q^*(i)| = 0 \quad \textit{for all} \quad i \le \lfloor\lambda\rfloor,$$

$$\limsup_{n\to\infty} \sup_{t\in[\bar{t}_{\text{eq}},T]} |q_n(t,i) - q^*(i)|e^{t-\bar{t}_{\text{eq}}} \le c \quad \textit{for all} \quad i \ge \lceil\lambda\rceil,$$

almost surely for all $T \ge \bar{t}_{\text{eq}}$; and c may be expressed in terms of α, λ, and $u(0)$ alone.

5. Simulations

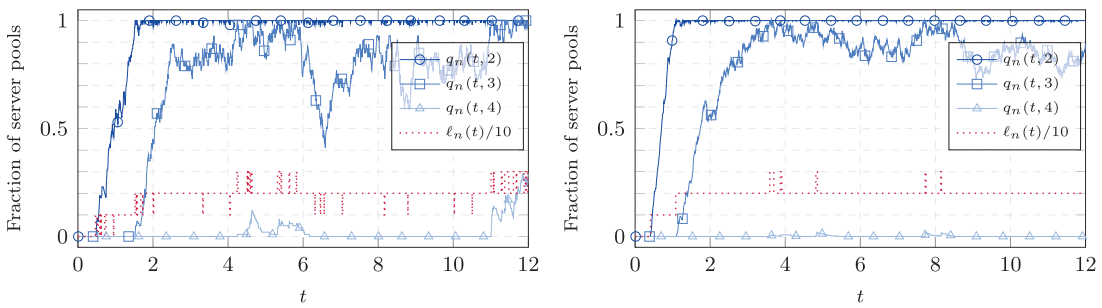
In this section, we investigate the threshold policy through several numerical experiments.[2] The first set of experiments evaluates the fluid and diffusion approximations suggested by our theoretical results. Next, we examine the user-perceived performance of several dispatching policies, including the threshold policy; we present a histogram of the empirical distribution of the fraction of resources received by a user in stationarity. Finally, we show that the threshold policy copes effectively with highly variable demand patterns.

First, we investigate how large n must be for ℓ_n to settle, as stated in Theorem 5 for the fluid limit. To this end, we consider systems with $\lambda = 2.9$ and different values of n; in Figure 3 we plot trajectories of two of these systems. In the system with $n = 100$, the threshold hovers around the optimal value, oscillating between $\lfloor\lambda\rfloor - 1 = 1$ and $\lceil\lambda\rceil = 3$. In the system with $n = 400$, the threshold stays at $\lfloor\lambda\rfloor = 2$ most of the time, with sporadic and brief excursions to $\lceil\lambda\rceil = 3$. The oscillations disappear completely when $n = 500$; this is not shown in Figure 3 but can be checked in the other experiments presented in this section, which correspond to $n = 500$.

Besides the number of server pools, the convergence of ℓ_n depends on the fractional part of λ. Specifically, the larger $\lambda - \lfloor\lambda\rfloor$ is, the larger n will have to be for ℓ_n to settle; observe that $\lambda - \lfloor\lambda\rfloor$ is relatively large in Figure 3. To understand this behavior, suppose that ℓ_n has reached $\lfloor\lambda\rfloor$; then Theorem 3 suggests that $q_n(\lceil\lambda\rceil)$ will oscillate around $\lambda - \lfloor\lambda\rfloor$ with deviations of order $1/\sqrt{n}$. If $\lambda - \lfloor\lambda\rfloor$ is large, then n must also be large in order to prevent $q_n(\lceil\lambda\rceil)$ from reaching one with high probability.

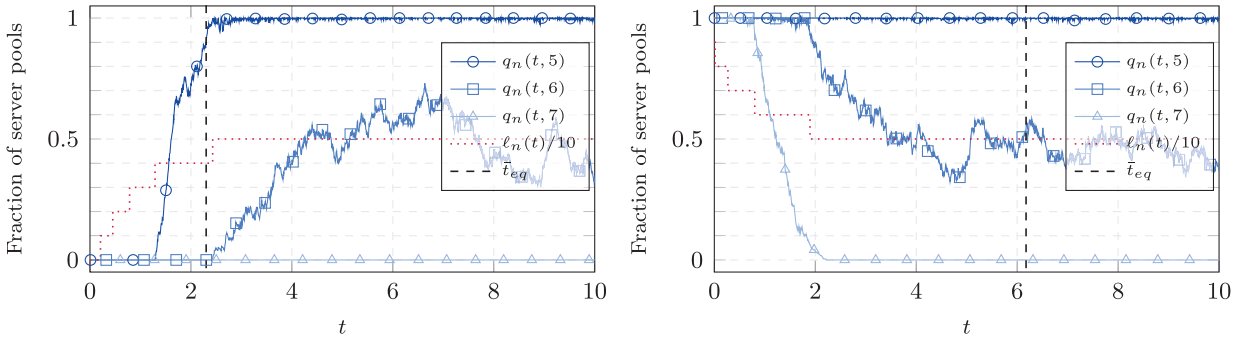
We proceed to assess the upper bound (15) for the time until the threshold settles. Figure 4 shows trajectories of two systems with $\lambda = 5.5$, $n = 500$, and different initial conditions; in both cases, $\alpha = 0.93$ satisfies (14). Whereas in the initially empty system ℓ_n settles at $\lfloor\lambda\rfloor = 5$ almost exactly at $\bar{t}_{\text{eq}}$, in the initially overloaded system, ℓ_n settles several units of time before $\bar{t}_{\text{eq}}$. In both cases, the threshold settles quickly in less than the average time needed to execute three tasks. In Figure 4, we can also see the trajectories of $q_n(\lfloor\lambda\rfloor)$ and $q_n(\lceil\lambda\rceil)$ after the threshold has settled at the optimal value. The former trajectories present very subtle oscillations, whereas the latter exhibit more

Figure 3. (Color online) Evolution of ℓ_n over Time When $\lambda = 2.9$ and $\alpha = 0.97$



Notes. On the left, $n = 100$, and the threshold fluctuates between $\lfloor\lambda\rfloor - 1$ and $\lceil\lambda\rceil$. On the right, $n = 400$, and the threshold stays at $\lfloor\lambda\rfloor$ most of the time.

Figure 4. (Color online) Time Until the Threshold Settles When $\lambda = 5.5$, $\alpha = 0.93$, and $n = 500$



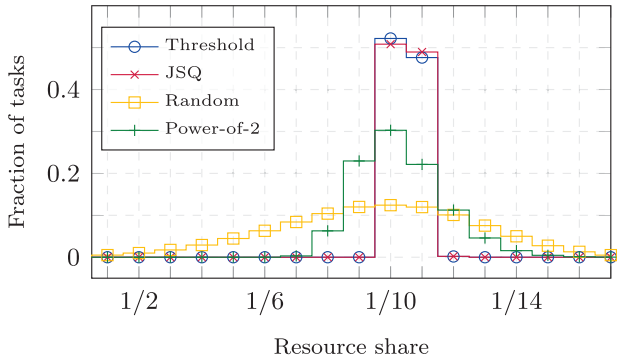
Notes. The system on the left is initially empty, and (15) is tight. On the right, all server pools have nine tasks initially, and (15) is not tight.

variability. This coincides with Theorem 3, which suggests that the deviations of $q_n(\lfloor\lambda\rfloor)$ from one are of order $\log(n)/n$ and that those of $q_n(\lceil\lambda\rceil)$ from $\lambda - \lfloor\lambda\rfloor$ are of order $1/\sqrt{n}$.

We now evaluate the quality of service experienced by users. To this end, we ran long simulations for several dispatching policies in stationarity, and we computed the empirical distribution of the fraction of resources received by an arbitrary user. We assumed the resources of any given server pool were equitably distributed among the tasks sharing it and we computed at each instant of time the number of tasks receiving a certain fraction of resources. We then integrated these quantities over time and normalized them to add up to one, as shown in Figure 5. The policy that assigns tasks to server pools chosen uniformly at random exhibits the largest variance, with some tasks receiving all the resources of a given server pool and some others contending for resources with as many as 15 tasks. Users are treated more fairly when tasks are sent to the least congested of two server pools chosen uniformly at random, but still some tasks share a server pool with as many as 13 other tasks. Finally, the fraction of resources assigned to tasks is either 1/10 or 1/11 for virtually all tasks when the threshold policy or JSQ are used. For these two policies, the load is evenly distributed and tasks are thus treated fairly; no users experience an inferior quality of service. In particular, the Schur-concave utilities mentioned in Section 1, which measure the overall experienced performance, are maximal.

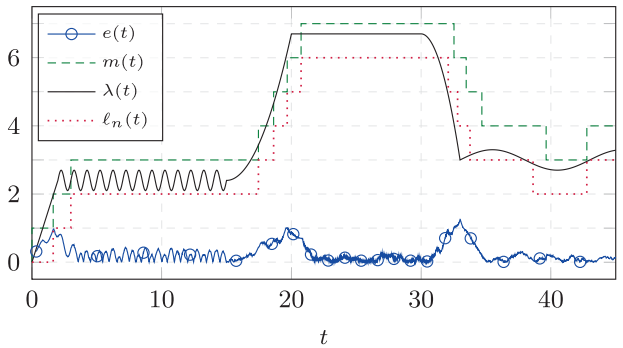
We conclude by investigating the response of the threshold policy to highly variable demand patterns. The trajectories depicted in Figure 6 correspond to a system where λ is time-varying and α was chosen according to (13) for $\lambda_{\max} = 10$. We observe that the system copes effectively with drastic and abrupt load variations, as the ones occurring around $t = 0$, 20, 30; in all these cases, ℓ_n quickly reaches the new optimal value. Also, the small but swift load fluctuations within the interval [2, 15] do not move the threshold away from the optimal value, which remains constant along the entire interval. Finally, ℓ_n adjusts to the slow load variations within [33, 45], which, in this case, result in changes of the optimal value. Along with the threshold, $m(t) = \max\{i : q_n(t,i) > 0\}$ and the quadratic error

Figure 5. (Color online) User-Perceived Performance Under Different Dispatching Rules



Notes. All policies were simulated with $\lambda = 10.5$ and $n = 500$. Simulation for $n = 500$ and $\alpha = 0.91$ Chosen from (13) for $\lambda_{\max} = 10$.

Figure 6. (Color online) Response of the Threshold Policy to a Highly Variable Demand Pattern



$e(t) = \|q_n(t) - q^*(t)\|_2$ are plotted; here $q^*(t)$ is computed in terms of $\lambda(t)$ from (1). We observe that $m(t) = \lceil \lambda(t) \rceil$ most of the time and that $e(t)$ is generally small, with its peak values coinciding with the most drastic changes of $\lambda(t)$; that is, concentrations of tasks at individual server pools are avoided and the loads are close to balanced most of the time.

6. Conclusions

In this paper, we examined a self-learning threshold-based policy for dispatching tasks in a system of parallel server pools, with the aim of balancing the total number of tasks evenly. We proved that this policy achieves the latter objective asymptotically, both on the fluid and the diffusion scales. These results were supported by several numerical experiments that showed that the optimality properties can already be seen in systems with a few hundred server pools, and we presented additional simulations that indicate that our policy copes effectively with highly variable demand patterns. Also, we provided a token-based implementation that involves at most two messages per task and storage of only little state information at the dispatcher, in the form of two tokens per server pool.

Our self-learning scheme adjusts the threshold over time to find an optimal value and, when an update occurs, the new threshold must be known by the dispatcher and all server pools. Synchronizing the threshold value of these entities could introduce a considerable communication burden during threshold updates. Although these updates occur only a few times before the threshold settles, a distributed procedure for adjusting the threshold could reduce the communication burden, and the design of such a procedure is a topic for future research. Another possible line of future inquiry is to study whether our adaptive load-balancing policy can be combined with mechanisms for adjusting the number of server pools over time, particularly in cloud computing deployments where the processing capacity is elastic. Specifically, the overall quality of service can be unacceptable if the offered load is too large, even if tasks are evenly distributed. In order to overcome this situation, the capacity of the system could be increased by deploying more server pools.

Throughout the paper, we considered systems of infinite server pools. Our self-learning threshold-based policy can be extended to scenarios where the maximum number of flows in each of the server pools is subject to a finite upper bound B, by imposing that the threshold cannot be increased beyond $B-1$. Unlike the infinite-server setting, this model allows to consider heavy-traffic regimes where the difference between the total number of servers nB and the arrival rate λ_n of the nth system grows sublinearly with n. A topic of future research is the blocking probability of our threshold policy in these regimes, particularly in the Halfin-Whitt regime where $(\lambda_n - nB)/\sqrt{n}$ has a finite limit as n grows to infinity.

Additional directions of future research concern extensions of our model to systems of heterogeneous server pools. One source of heterogeneity could be the way in which the service rate of tasks depends on the number of concurrent tasks at each server pool; in our model, the service rate is equal to the number of tasks. The fluid limits derived in this paper carry over to the general setting, but establishing that the learning rule for adjusting the threshold always reaches an equilibrium requires nontrivial additional arguments, since our proof relies on the property of infinite-server systems that the total number of tasks is not affected by the assignment decisions. Another source of heterogeneity could be the utility function u mentioned in Section 2.1 as a proxy for measuring quality of service. It would be interesting to investigate how the results in this paper generalize to systems where the experienced performance depends on the type of server pool, and thus evenly balancing the load may not maximize the overall quality of service.

Endnotes

[1] All the data depicted in the plots presented in this paper, and the code used to generate these data, are available at https://github.com/diegogolds/self-learning-threshold-policy.

[2] All the data depicted in the plots presented in this section, and the code used to generate these data, are available at https://github.com/diegogolds/self-learning-threshold-policy.

References

Baccelli F, Kauffmann B, Veitch D (2009) Inverse problems in queueing theory and internet probing. *Queueing Systems* 63:59.

Badonnel R, Burgess M (2008) Dynamic pull-based load balancing for autonomic servers. *Proc. 2008 IEEE Network Oper. Management Sympos.* (IEEE, Piscataway, NJ), 751–754.

Benameur N, Fredj SB, Oueslati-Boulahia S, Roberts JW (2002) Quality of service and flow level admission control in the Internet. *Comput. Networks* 40(1):57–71.

Billingsley P (2013) *Convergence of Probability Measures* (Wiley, New York).

Bonald T, Jonckheere M, Proutière A (2004) Insensitive load balancing. *ACM SIGMETRICS Performance Evaluation Rev.* 32(1): 367–377.

Bramson M (1998) State space collapse with application to heavy traffic limits for multiclass queueing networks. *Queueing Systems* 30(1-2):89–140.

Comte C (2019) Dynamic load balancing with tokens. *Comput. Commun.* 144:76–88.

Defraeye M, van Nieuwenhuyse I (2016) Staffing and scheduling under nonstationary demand for service: A literature review. *Omega* 58:4–25.

Ephremides A, Varaiya P, Walrand J (1980) A simple dynamic routing problem. *IEEE Trans. Automatic Control* 25(4):690–693.

Gamarnik D, Tsitsiklis JN, Zubeldia M (2018) Delay, memory, and messaging tradeoffs in distributed service systems. *Stochastic Systems* 8(1):45–74.

Goldsztajn D, Ferragut A, Paganini F (2018a) Feedback control of server instances for right sizing in the cloud. *Proc. 56th Annual Allerton Conf. Comm. Control Comput*. (IEEE, Piscataway, NJ), 749–756.

Goldsztajn D, Ferragut A, Paganini F, Jonckheere M (2018b) Controlling the number of active instances in a cloud environment. *ACM SIGMETRICS Performance Evaluation Rev.* 45(3):15–20.

Gupta V, Harchol-Balter M (2009) Self-adaptive admission control policies for resource-sharing systems. *ACM SIGMETRICS Performance Evaluation Rev.* 37(1):311–322.

Horváth IA, Scully Z, van Houdt B (2019) Mean field analysis of join-below-threshold load balancing for resource sharing servers. *Proc. ACM Measurement Anal. Comput. Systems* 3(3):57.

Jonckheere M, Prabhu BJ (2016) Asymptotics of insensitive load balancing and blocking phases. *Proc. 2016 ACM SIGMETRICS Internat. Conf. Measurement Modeling Comput. Sci.* (ACM, New York), 311–322.

Karthik A, Mukhopadhyay A, Mazumdar RR (2017) Choosing among heterogeneous server clouds. *Queueing Systems* 85:1–29.

Key P, Massoulié L, Bain A, Kelly F (2004) Fair internet traffic integration: Network flow models and analysis. *Ann. Télécomm.* 59: 1338–1352.

Lu Y, Xie Q, Kliot G, Geller A, Larus JR, Greenberg A (2011) Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation* 68(11):1056–1071.

Menich R, Serfozo RF (1991) Optimality of routing and servicing in dependent parallel processing systems. *Queueing Systems* 9(4): 403–418.

Mitzenmacher M (2001) The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distributed Systems* 12(10): 1094–1104.

Mukherjee D, Borst SC, van Leeuwaarden JS, Whiting PA (2016) Universality of load balancing schemes on the diffusion scale. *J. Appl. Probab.* 53(4):1111–1124.

Mukherjee D, Borst SC, van Leeuwaarden JS, Whiting PA (2020) Asymptotic optimality of power-of-*d* load balancing in large-scale systems. *Math. Oper. Res.* 45(4):1535–1571.

Mukherjee D, Dhara S, Borst SC, van Leeuwaarden JS (2017) Optimal service elasticity in large-scale distributed systems. *Proc. ACM Measurement Anal. Comput. Systems* 1(1):25.

Mukhopadhyay A, Mazumdar RR, Guillemin F (2015a) The power of randomized routing in heterogeneous loss systems. Proc. 27th Internat. Teletraffic Congress (IEEE, Piscataway, NJ), 125–133.

Mukhopadhyay A, Karthik A, Mazumdar RR, Guillemin F (2015b) Mean field and propagation of chaos in multi-class heterogeneous loss models. *Performance Evaluation* 91:117–131.

Sparaggis PD, Towsley D, Cassandras C (1993) Extremal properties of the shortest/longest non-full queue policies in finite-capacity systems with state-dependent service rates. *J. Appl. Probab.* 30(1):223–236.

Stolyar AL (2015) Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Systems* 80(4):341–361.

Turner SR (1998) The effect of increasing routing choice on resource pooling. *Probab. Engrg. Inform. Sci.* 12(1):109–124.

van der Boor M, Borst SC, van Leeuwaarden JS, Mukherjee D (2018) Scalable load balancing in networked systems: A survey of recent advances. Preprint, submitted June 14, https://arxiv.org/abs/1806.05444.

Vvedenskaya ND, Dobrushin RL, Karpelevich FI (1996) Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Inform.* 32(1):20–34.

Wierman A, Andrew LL, Tang A (2012) Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation* 69(12):601–622.

Winston W (1977) Optimality of the shortest line discipline. *J. Appl. Probab.* 14(1):181–189.

Xie Q, Dong X, Lu Y, Srikant R (2015) Power of *d* choices for large-scale bin packing: A loss model. *ACM SIGMETRICS Performance Evaluation Rev.* 43(1):321–334.

Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. *Proc. 36th Annual IEEE Sympos. Foundations Computer Sci.* (IEEE, Piscataway, NJ), 374–382.

Zhou X, Tan J, Shroff N (2019) Heavy-traffic delay optimality in pull-based load balancing systems: Necessary and sufficient conditions. *ACM SIGMETRICS Performance Evaluation Rev.* 47(1):5–6.

Zhou X, Wu F, Tan J, Sun Y, Shroff N (2017) Designing low-complexity heavy-traffic delay-optimal load balancing schemes: Theory to algorithms. *Proc. ACM Measurement Anal. Comput. Systems* 1(2):39.