Zero-wait load balancing with sparse messaging

van der Boor, Mark; Zubeldia, Martin; Borst, Sem

DOI
10.1016/j.orl.2020.04.006
Publication date
2020
Document Version
Final published version
Published in
Operations Research Letters
License
CC BY

Operations Research Letters 48 (2020) 368–375

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Zero-wait load balancing with sparse messaging

Mark van der Boor [a,*], Martin Zubeldia [a,b], Sem Borst [a]

[a] Eindhoven University of Technology, The Netherlands
[b] Korteweg-de Vries Institute, University of Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 20 December 2019
Received in revised form 3 March 2020
Accepted 10 April 2020
Available online 20 April 2020

Keywords:
Load balancing
Communication overhead
Queueing delay
Job assignment

ABSTRACT

A key challenge in designing load balancing strategies is to achieve low delay in large-scale systems while only using minimal communication overhead. Motivated by these issues, we introduce a novel 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.

1. Introduction

Well-designed load balancing strategies provide an effective mechanism for improving delay performance and achieving efficient resource utilization in parallel-processing systems such as cloud networks and data centers. Besides these typical performance criteria, communication overhead and implementation complexity have emerged as equally crucial concerns due to the immense numbers of servers in large-scale deployments. Such scalability challenges have fueled an urgent interest in load balancing algorithms that yield excellent delay performance while only requiring low implementation overhead. Motivated by these issues, we introduce and analyze a novel concept to achieve asymptotically negligible queueing delay in a many-server regime with a minimal amount of information exchange. Before further explaining the concept, we first give an overview of relevant literature and highlight the main developments.

We focus on a basic scenario of N parallel identical servers and a single dispatcher where jobs arrive that must immediately be forwarded to one of the servers. If the service requirements are exponentially distributed, and the service discipline at each server is oblivious to the actual service requirements (e.g. FIFO), then the classical Join-the-Shortest-Queue (JSQ) policy has strong stochastic optimality properties [3,13]. In order to execute the JSQ policy, however, the dispatcher requires instantaneous knowledge of the queue lengths at all the servers, which may involve a substantial communication burden, and may not be scalable for large numbers of servers.

The latter issue has spurred a strong interest in so-called JSQ(d) strategies, where the dispatcher assigns incoming jobs to a server with the shortest queue among d servers selected uniformly at random. This involves an exchange of $2d$ messages per job (assuming $d \geq 2$), and thus drastically reduces the communication overhead compared to the full JSQ policy when the number of servers is large. At the same time, even a value as small as $d = 2$ yields significant performance improvements in the many-server regime $N \to \infty$ compared to a purely random assignment scheme ($d = 1$) [8,12].

However, JSQ(d) strategies lack the ability of the conventional JSQ policy to drive the queueing delay to zero as $N \to \infty$ for any finite value of d. In contrast, if d grew with N, making it possible to drive queueing delay to zero [9], the communication overhead would grow unboundedly. This is in fact inevitable, since results in the seminal paper [5] show that it is fundamentally impossible to achieve a vanishing queueing delay with a finite communication overhead per job, *unless* memory is available at the dispatcher to store state information.

The latter is exactly what is done in the so-called Join-the-Idle-Queue (JIQ) scheme [2,7], where servers advertise their availability by sending a 'token' to the dispatcher whenever they become idle, thus generating at most one message per job. The dispatcher assigns incoming jobs to an idle server as long as tokens are outstanding, or to a uniformly at random selected server otherwise. 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 [4,11]. Scaling results for the JIQ scheme in an asymptotic regime where the load approaches one as the system grows large may be found in [6,10].

As mentioned above, the JIQ scheme uses at most one message per job since a server only sends a token when it becomes idle

* Correspondence to: P.O. Box 513, 5600 MB, Eindhoven, The Netherlands.
E-mail address: m.v.d.boor@tue.nl (M. van der Boor).

https://doi.org/10.1016/j.orl.2020.04.006

upon a job completion. The vanishing queueing delay achieved in the many-server limit indicates that actually almost any job completion leaves the server idle, and means that the number of messages per job in fact approaches one. Now 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 one has access to the job sizes, as in [1], 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. See also [5,14].

Our contributions. In the present paper we debunk the above-mentioned notion and introduce a scheme that allows for vanishing queueing delays and that uses 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.

More specifically, we introduce the **Join-the-Open-Queue (JOQ) scheme**, where 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 T 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 T. In particular, the waiting probability under the JOQ scheme is upper bounded by that under the JIQ scheme with the inflated service requirements. We then leverage the above-mentioned results for the JIQ scheme to demonstrate that, for any load below a certain threshold $\lambda^* < 1$, vanishing queueing delay is also achieved by the JOQ scheme for a suitable choice of T with strictly less than one message per job. It turns out that there is a range of values for T, 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 the paper is organized as follows. In Section 2 we present a detailed model description and algorithm specification, and we summarize the main contribution of the paper. In Section 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 4 we introduce the system design where jobs are distributed over servers executing JOQ and servers using JIQ. In Section 5 we provide numerical simulations and discuss various broader issues and observations. Section 6 concludes and mentions several future research avenues.

2. Model and algorithm

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 T 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 $T\lceil x/T\rceil \geq x$ 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 T 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 a server chosen uniformly at random.*

Note that the JOQ algorithm is emulating the JIQ algorithm but using open/closed servers instead of idle/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 issues for the JOQ algorithm can be found in Section 5, where we use simulation results to put some of the choices into perspective.

3. Main results

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. In particular, we will show that:

(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.

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 T the JOQ algorithm has vanishing queueing delay, and provides an expression for the limit of the expected number of messages per job.

Theorem 1. *For every T such that $\lambda\mathbb{E}[T\lceil X/T\rceil] < 1/2$, the following properties hold for the JOQ algorithm with that value of T:*

(i) it has vanishing queueing delay,
(ii) the expected number of messages per job converges to $\mathbb{E}[\lfloor X/T\rfloor]$, as $N \to \infty$.

When $\lambda < 1/2$, a value of $T > 0$ exists such that $\lambda\mathbb{E}[T\lceil X/T\rceil] < 1/2$ regardless of the job size distribution, since $\lambda\mathbb{E}[T\lceil X/T\rceil] \to \mathbb{E}[X]$ as $T \downarrow 0$.

Proof of Theorem 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 T, 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 $T\lceil X/T\rceil$. Since the expected size of a job in the virtual queues is $\mathbb{E}[T\lceil X/T\rceil]$, their load is $\lambda\mathbb{E}[T\lceil X/T\rceil] < 1/2$. Thus, Theorem 2 in [4] 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 [4], 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 $[0, T\lceil x/T\rceil)$. During this interval of time, messages are sent every T time units (excluding 0), which results in $\lfloor x/T\rfloor$ 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 T time units while the virtual server is busy, similarly to when jobs are sent to open servers. In total, this results in $1 + \lfloor x/T\rfloor$ 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 [4]). 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 $p_o(N)\mathbb{E}[\lfloor X/T\rfloor] + [1 - p_o(N)](1 + \mathbb{E}[\lfloor X/T\rfloor])$. Since Theorem 2 in [4] also implies that $p_o(N) \to 1$ as $N \to \infty$, the expected number of messages per job converges to $\mathbb{E}[\lfloor X/T\rfloor]$, as $N \to \infty$. □

In light of Theorem 1, we would require $\lambda\mathbb{E}[T\lceil X/T\rceil] < 1/2$ and $\mathbb{E}[\lfloor X/T\rfloor] < 1$ 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 T this is accomplished when $\lambda < 1/4$.

Corollary 1. *Suppose that $\lambda < 1/4$. For all T such that $1 < T < 1/(2\lambda) - 1$, 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 T always exists, since $1 < 1/(2\lambda) - 1$ when $\lambda < 1/4$.

Proof of Corollary 1. Corollary 1 follows from Theorem 1 since $\lambda\mathbb{E}[T\lceil X/T\rceil] \le \lambda(T + \mathbb{E}[X]) = \lambda(T+1) < \lambda(1/(2\lambda) - 1 + 1) = 1/2$ and $\mathbb{E}[\lfloor X/T\rfloor] \le \mathbb{E}[X]/T = 1/T < 1$. □

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 $T = T(\lambda)$ needs to be adjusted appropriately given the load λ.

Corollary 2. *When the value of T in the JOQ algorithm is chosen depending on λ such that $T(\lambda) < 1/(2\lambda) - 1$ and $T(\lambda) \to \infty$ 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 1 guarantees a vanishing queueing delay since $T(\lambda) < 1/(2\lambda) - 1$ for any λ, so also more specifically for the limit when $\lambda \downarrow 0$. Moreover, Theorem 1 states that the expected number of messages per job in the many-server limit converges to $\mathbb{E}[\lfloor X/T\rfloor]$, with $\mathbb{E}[\lfloor X/T(\lambda)\rfloor] \le \mathbb{E}[X]/T(\lambda) = 1/T(\lambda) \downarrow 0$ as $\lambda \downarrow 0$. □

Another desirable property of the JOQ algorithm is that, although Corollary 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 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 $T = 1/(4\lambda)$ has vanishing queueing delay, and the expected number of messages per job is less than one in the many-server limit.*

Proof of Corollary 3. 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 $\frac{1-F((k+1)T)}{1-F(kT)} \leq \frac{1-F(kT)}{1-F((k-1)T)}$. Successively applying this inequality gives $\frac{1-F((k+1)T)}{1-F(kT)} \leq 1-F(T)$ for any integer k. This leads to the inequality

$$\begin{aligned}\mathbb{P}(X > kT) = 1 - F(kT) &= \prod_{i=0}^{k-1} \frac{1-F((i+1)T)}{1-F(iT)} \\ &\leq (1-F(T))^k = (\mathbb{P}(X > T))^k .\end{aligned} \tag{1}$$

Using this inequality, it follows that

$$\begin{aligned}\mathbb{E}[T\lceil X/T\rceil] = T\sum_{k=0}^{\infty}\mathbb{P}(X > kT) &\leq T\sum_{k=0}^{\infty}(\mathbb{P}(X > T))^k \\ &= \frac{T}{1-\mathbb{P}(X > T)} < \frac{1}{2\lambda},\end{aligned} \tag{2}$$

where in the last inequality we used that $T = 1/(4\lambda)$ and that $\mathbb{P}(X > 1/(4\lambda)) < 1/2$. This allows us to use Theorem 1 showing that the queueing delay vanishes. Furthermore, the expected number of messages per job is

$$\begin{aligned}\mathbb{E}[\lfloor X/T\rfloor] = \sum_{k=1}^{\infty}\mathbb{P}(X > kT) &\leq \sum_{k=1}^{\infty}(\mathbb{P}(X > T))^k \\ &= \frac{\mathbb{P}(X > T)}{1-\mathbb{P}(X > T)} < 1,\end{aligned} \tag{3}$$

where in the last inequality we once again used that $T = 1/(4\lambda)$ 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 4. *Suppose that the job size X is exponentially distributed and that $\lambda < 1/\ln(16)$. Then, there exists some T such that the JOQ algorithm with that value of T has vanishing queueing delay, and the expected number of messages per job is less than one in the many-server limit.*

Proof of Corollary 4. First note that, for exponential job sizes, (1) holds with equality. Thus, Theorem 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 $\mathbb{E}[T\lceil X/T\rceil] = \frac{T}{1-\mathbb{P}(X>T)} = \frac{T}{1-e^{-T}} < \frac{1}{2\lambda}$ and $\mathbb{E}[\lfloor X/T\rfloor] = \frac{\mathbb{P}(X>T)}{1-\mathbb{P}(X>T)} = \frac{e^{-T}}{1-e^{-T}} < 1$. It can be checked that there exists a $T > 0$ such that these two inequalities are satisfied as long as $\lambda < 1/\ln(16)$. □

Remark 1. When the job size distribution is exponential, it is optimal to take the update epochs equidistant, see Appendix A.

Remark 2. In [4] 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 1 would hold for all $\lambda < 1$ and for all T such that $\lambda\mathbb{E}[T\lceil X/T\rceil] < 1$. This decreases the communication overhead since T can be chosen larger. Moreover, such a stronger result would also improve the conditions stated in Corollaries 1 and 3. For example, Corollary 1 would hold for all $\lambda < 1/2$ and for all T such that $1 < T < 1/\lambda - 1$.

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).

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 T (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; [11]), or 1/2 otherwise (case 2; [4]).

Theorem 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 T are chosen such that $p > 0$, (a) $\lambda(1-p)/(1-f) < \rho$, (b) $\lambda p/f < 1/4$ and (c) $1 < T < f/(2\lambda p) - 1$. Such values for f, p and T always exist.*

Proof of Theorem 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 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 T 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 T 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 $\lambda < 1/2$. □

4.2. Choosing the system parameters

Since the parameters f, p, and T 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.

$$\begin{aligned} \inf_{p,f,T} \quad & 1-p+p\mathbb{E}[\lfloor X/T \rfloor] \\ \text{s.t.} \quad & \lambda pT\mathbb{E}[\lceil X/T \rceil] < f/2, \quad \lambda(1-p) < \rho(1-f), \\ & 0 \le f \le 1, \quad 0 \le p \le 1, \quad T \ge 0, \end{aligned} \tag{4}$$

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

$$\lambda pT\mathbb{E}[\lceil X/T \rceil] \le f/2-\epsilon \qquad \text{and} \qquad \lambda(1-p) \le \rho(1-f)-\epsilon.$$

Furthermore, our simulations (Section 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 (4) would need to be

$$\lambda pT\mathbb{E}[\lceil X/T \rceil] < f \qquad \text{and} \qquad \lambda(1-p) < (1-f). \tag{5}$$

For exponential job sizes, it can be shown that the infimum of the problem given by (4) is attained when the parameters are selected as

$$\begin{cases} (i)\ T \text{ such that } 1-e^{-T} = 2\lambda T, p = 1, f = 1, \\ (ii)\ T \text{ s.t. } 2e^T - 4T = 3, p = \frac{(1-\lambda)(e^T-1)}{\lambda(1-e^T+2Te^T)}, f = 1-\lambda(1-p), \end{cases} \tag{6}$$

when (i) $\lambda \le \lambda^*$ or (ii) $\lambda > \lambda^*$, where λ^* solves $\frac{(1-\lambda)(e^T-1)}{\lambda(1-e^T+2Te^T)} = 1$ when $2e^T - 4T = 3$. These parameters cause equality in some of the equations where a strict inequality is needed. The derivation of (6) is presented in Appendix B. The solution value is shown in Fig. 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 (6)), and where this also holds true for all $\lambda < 1$ (which is widely believed but remains to be proven). See Appendix B for further details on this case.

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.

5.1. The choice of T

We investigate the performance of the JOQ algorithm. We therefore plot the probability of queueing as a function of T, for $\lambda = 0.3$, in Fig. 2. The thin blue dashed line represents the number of messages JIQ uses, and the thin red dashed line represents the probability of queueing for the random dispatching policy.

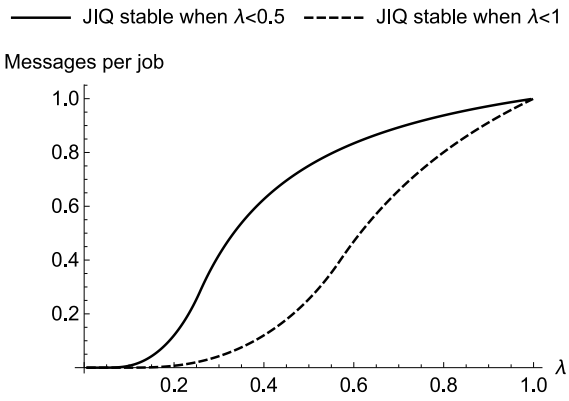

Fig. 1. Numerical computation: the average number of messages per job as a function of the arrival rate.

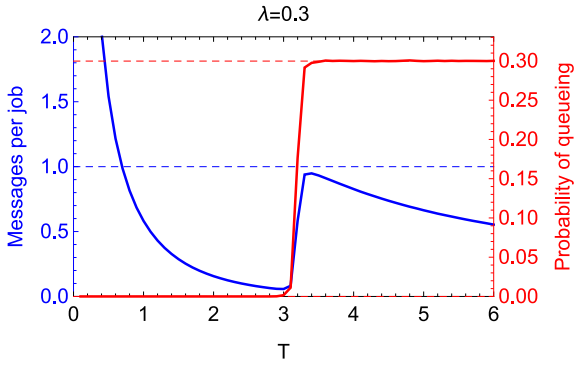

Fig. 2. Probability of queueing and average number of messages per job as a function of T ($\lambda = 0.3$).

In this figure, it is clear that the probability of queueing and the message rate are both extremely low when T is chosen to be somewhere between 2 and 3. As expected, when T 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 4.2 we described a method to determine the optimal value of T, as part of the solution of the optimization problem given in (4). In particular, for $\lambda = 0.3$ this would lead to $T \approx 1.51$. However, in Fig. 2, one clearly observes that larger values of T 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), then the optimal value of T would be approximately 3.20. This matches with the phase transition in the performance observed at this value of T, 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 T 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 T time units.

More extensive results for $\lambda = 0.5$ follow in the next subsection. Similar observations hold for other values of λ as well, and hence we omitted these figures. Note that for these larger values of λ, the exact choice of T becomes more important. The fact that

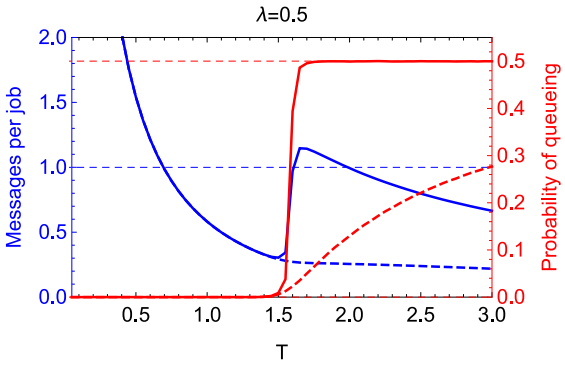

Fig. 3. Probability of queueing and average number of messages per job as a function of T ($\lambda = 0.5$). The dashed lines represent the performance of the system without virtual queues.

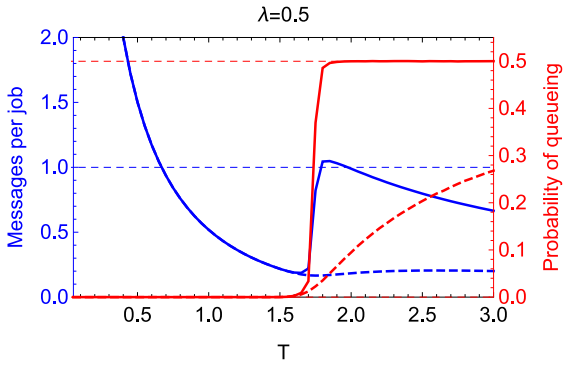

Fig. 5. Probability of queueing and average number of messages per job as a function of T ($\lambda = 0.5$), when the job size distribution has decreasing hazard rate ($Gamma(2, 2)$).

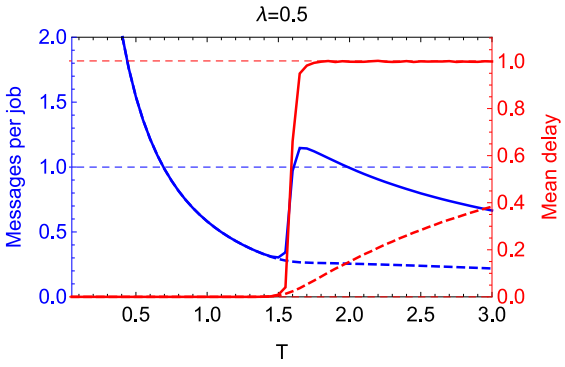

Fig. 4. Mean delay and average number of messages per job as a function of T ($\lambda = 0.5$). The dashed lines represent the performance of the system without virtual queues.

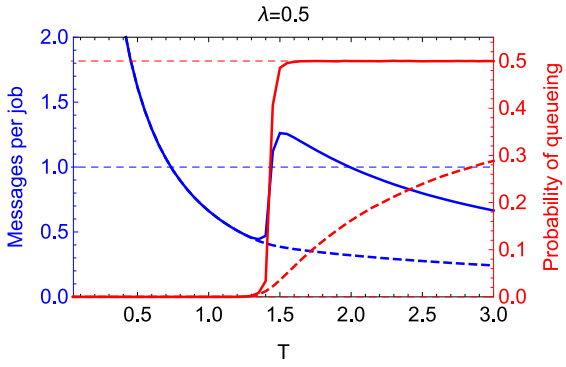

Fig. 6. Probability of queueing and average number of messages per job as a function of T ($\lambda = 0.5$), when the job size distribution has increasing hazard rate ($Gamma(1/2, 1/2)$).

the JOQ algorithm still achieves less than one message per job when $\lambda = 0.5$ is remarkable.

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 Figs. 3 and 4, 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 T. This is because the virtual queues are stable in both cases, as their load is less than one. However, when T is close to the 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 T is chosen larger than its optimal value.

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 Appendix A.

For non-exponential job sizes, the behavior of the mean delay and probability of queueing is quite different, as can be observed in Figs. 5 and 6. When the hazard rate is decreasing, larger values of T are acceptable for vanishing delays, which could be explained by observing that (2) is an upper bound for the relative load when the job size distribution has decreasing hazard rate, because of (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 T. The expected number of messages per job also seems to be larger.

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 would 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 servers would be eliminated as option to dispatch to.

An open question is whether the system design 'hybrid JOQ/JIQ' in Section 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.

Acknowledgment

The work in this paper is supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

Appendix A. Optimality of equidistant updates for exponential jobs

Let us relax the assumption that, under the JOQ algorithm, the time between update epochs is always the same T. For $i \geq 1$, let s_i be the time between the $(i-1)$-th and the ith 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.

$$\begin{aligned} \inf_{s \geq 0} \quad & \sum_{i=1}^{\infty} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) \\ s.t. \quad & \lambda \sum_{i=0}^{\infty} s_{i+1} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) \leq \rho. \end{aligned} \tag{7}$$

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 3. *Suppose that the job sizes are exponentially distributed. If s^* attains the infimum in (7), then there exists some $T > 0$ such that $s_k^* = T$, for all $k \geq 1$.*

Proof. We first obtain a sequence of implicit equations for s^*.

Lemma A.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*

$$C = \frac{\sum_{i=k}^{\infty} s_{i+1}^* f\left(\sum_{j=1}^{i} s_j^*\right) - \mathbb{P}\left(X > \sum_{j=1}^{k-1} s_j^*\right)}{\sum_{i=k}^{\infty} f\left(\sum_{j=1}^{i} s_j^*\right)} \tag{8}$$

for all $k \geq 1$, and

$$C = s_{k+1}^* - \frac{\mathbb{P}\left(\sum_{j=1}^{k-1} s_j^* < X \leq \sum_{j=1}^{k} s_j^*\right)}{f\left(\sum_{j=1}^{k} s_j^*\right)}, \tag{9}$$

for all $k \geq 2$.

Proof. Since $\mathbb{P}(X > x)$ is convex, (7) is an infinite-dimensional convex optimization problem in s. Now consider the Lagrangian

$$\begin{aligned} \mathcal{L}(s, y) = & \sum_{i=1}^{\infty} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) \\ & - y\left[\lambda \sum_{i=0}^{\infty} s_{i+1} \mathbb{P}\left(X > \sum_{j=1}^{i} s_j\right) - \rho\right]. \end{aligned}$$

Let (s^*, y^*) be an optimal solution. For every $k \geq 1$, since $s_k^* = 0$ cannot be optimal, we must have

$$\begin{aligned} \frac{\partial \mathcal{L}(s^*, y^*)}{\partial s_k^*} = & -\sum_{i=k}^{\infty} f\left(\sum_{j=1}^{i} s_j^*\right) \\ & - y^* \lambda \left[-\sum_{i=k}^{\infty} s_{i+1}^* f\left(\sum_{j=1}^{i} s_j^*\right) + \mathbb{P}\left(X > \sum_{j=1}^{k-1} s_j^*\right)\right] \\ = & 0. \end{aligned}$$

Since $f(x) > 0$ for all $x \geq 0$, we have $y^* > 0$. Thus, rearranging terms we obtain Eq. (8) with $C = 1/(y^*\lambda) > 0$. Moreover, using that

$$\frac{\partial \mathcal{L}(s^*, y^*)}{\partial s_k^*} - \frac{\partial \mathcal{L}(s^*, y^*)}{\partial s_{k-1}^*} = 0$$

for all $k \geq 2$, and rearranging terms, we obtain Eq. (9). □

Combining (9) 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 $s_k^* = T$ for all $k \geq 2$, where T is the unique solution of

$$T = C + 1 - e^T.$$

On the other hand, combining (8) for the case $k = 2$, with the fact that the job sizes are exponentially distributed and that $s_k^* = T$ for all $k \geq 2$, we obtain

$$T = C + \left(1 - e^T\right) e^{T - s_1^*}.$$

Thus, we have $s_1^* = T$. □

Appendix B. Solution of the minimization problem

The minimization problem (4) when job sizes are exponentially distributed looks as follows.

$$\begin{aligned} \inf_{p,f,T} \quad & 1 - p + p\frac{e^{-T}}{1 - e^{-T}} \\ \text{s.t.} \quad & \lambda p T/(1 - e^{-T}) < f/2, \quad \lambda(1-p) < 1 - f, \\ & 0 \leq f \leq 1, \quad 0 \leq p \leq 1, \quad T \geq 0. \end{aligned} \tag{10}$$

Note that the objective function is decreasing in T, so T should be as large as possible. The first condition then shows that, in order to choose T 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

$$T/(1-e^{-T}) < \frac{1-(1-p)\lambda}{2\lambda p}, \tag{11}$$

where the right-hand side is decreasing in p. As we want to maximize T, we seek equality which gives

$$p(\lambda,T) = \frac{(1-\lambda)(e^T-1)}{\lambda(1-e^T+2Te^T)}, \tag{12}$$

where we note that this value may exceed 1, in which case we preserve the equality as much as possible and set $p(\lambda,T)=1$. Suppose $p(\lambda,T)<1$, in that case we choose T such that $2e^T-4T=3$ in order to minimize the objective function (this follows from differentiating the objective function and equating it to zero, after substituting (12)). However, with this value of T, $p(\lambda,T)$ 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 $p(\lambda,T)$ is strictly smaller than one, when T is chosen such that $2e^T-4T=3$. When λ is smaller than this λ^*, we choose $p=1$ after which the objective function is minimized by putting T such that (11) is met with equality which is the case when $1-e^{-T}=2\lambda T$.

B.1. JIQ Stable for $\lambda < 1$

If the stability result proved in [4] applies even when $\lambda < 1$, then (11), which corresponds to the first constraint in (10), becomes $\lambda pT/(1-e^{-T}) < f$. The optimal parameters for this problem are

$$\begin{cases} (i)\ T \text{ such that } 1-e^{-T}=\lambda T, p=1, f=1, \\ (ii)\ T \text{ such that } e^T-2T=1, p=\frac{(1-\lambda)(e^T-1)}{\lambda(1-e^T+Te^T)}, \\ \quad f=1-\lambda(1-p) \end{cases}$$

when (i) $\lambda \le \lambda^*$ or (ii) $\lambda > \lambda^*$, where λ^* solves $\frac{1-e^T+\lambda Te^T}{(1-e^T+Te^T)\lambda}=0$ when $e^T-2T=1$.

References

[1] J. Anselmi, Combining size-based load balancing with round-robin for scalable low latency, IEEE Trans. Parallel Distrib. Syst. (2019).

[2] R. Badonnel, M. Burgess, Dynamic pull-based load balancing for autonomic servers, in: IEEE Network Oper. and Management Symp., 2008, pp. 751–754.

[3] A. Ephremides, P. Varaiya, J. Walrand, A simple dynamic routing problem, IEEE Trans. Automat. Control 25 (4) (1980) 690–693.

[4] S.G. Foss, A.L. Stolyar, Large-scale join-idle-queue system with general service times, J. Appl. Probab. 54 (4) (2017) 995–1007.

[5] D. Gamarnik, J.N. Tsitsiklis, M. Zubeldia, Delay, memory and messaging tradeoffs in distributed service systems, in: Proc. SIGMETRICS '16, 2016, pp. 1–12.

[6] V. Gupta, N.S. Walton, Load balancing in the nondegenerate slowdown regime, Oper. Res. 67 (1) (2019) 281–294.

[7] Y. Lu, Q. Xie, G. Kliot, A. Geller, J.R. Larus, A. Greenberg, Join-idle-queue: a novel load balancing algorithm for dynamically scalable web services, Perform. Eval. 68 (11) (2011) 1056–1071.

[8] M. Mitzenmacher, The power of two choices in randomized load balancing, IEEE Trans. Parallel Distrib. Syst. 12 (10) (2001) 1094–1104.

[9] D. Mukherjee, S.C. Borst, J.S.H. van Leeuwaarden, P.A. Whiting, Universality of power-of-d load balancing in many-server systems, Stoch. Syst. 8 (4) (2018) 265–292.

[10] D. Mukherjee, S.C. Borst, J.S.H. van Leeuwaarden, P.A. Whiting, Universality of load balancing schemes on the diffusion scale, J. Appl. Probab. 53 (4) (2016) 1111–1124.

[11] A.L. Stolyar, Pull-based load distribution in large-scale heterogeneous service systems, Queueing Syst. 80 (4) (2015) 341–361.

[12] N.D. Vvedenskaya, R.L. Dobrushin, F.I. Karpelevich, Queueing system with selection of the shortest of two queues: An asymptotic approach, Probl. Pereda. Inf. 32 (1) (1996) 20–34.

[13] W. Winston, Optimality of the shortest line discipline, J. Appl. Probab. 14 (1) (1977) 181–189.

[14] L. Ying, R. Srikant, X. Kang, The power of slightly more than one sample in randomized load balancing, Math. Oper. Res. 42 (3) (2017) 692–722.