On the Performance Evaluation of Distributed Join-Idle-Queue Load Balancing with and without Token Withdrawals

Benny Van Houdt
Department of Computer Science
University of Antwerp, Belgium

Abstract

Distributed Join-Idle-Queue load balancing is known to achieve vanishing waiting times in the large-scale limit provided that the number of dispatchers remains fixed, while the number of servers tends to infinity. When the number of dispatchers m scales to infinity together with the number of servers n, such that $r = n/m$ remains fixed, the large-scale performance of Join-Idle-Queue load balancing is less clear as waiting times no longer vanish.

In this paper we first discuss some existing mean field models for distributed Join-Idle-Queue load balancing with $r = n/m$ fixed and explain why the well-known model introduced in [1] is not exact in the large-scale limit. The inexactness is caused by mixing two variants of distributed Join-Idle-Queue load balancing: a variant with and one without token withdrawals. Next we introduce mean field models for Join-Idle-Queue load balancing with and without token withdrawals, where an idle server places a token at a dispatcher with the shortest among d randomly chosen dispatchers.

The introduced mean field models in case of token withdrawals imply that for phase type distributed service times and a total job arrival rate of $\lambda n < n$, the response time of a job corresponds to that in a standard M/PH/1 queue with load λq_0. The value of q_0 can be determined numerically and depends on λ, r and d, but not on the job size distribution (apart from its mean). This simple behavior is lost if token withdrawals do not take place. For the models without withdrawals we develop fast numerical algorithms to determine the performance. We present simulation experiments that suggest that the unique fixed point of the introduced mean field models provides exact results in the large-scale limit.

1. Introduction

In traditional server farms jobs are distributed among the front end servers by a single hardware load balancer/dispatcher. While such a load balancer can support hundreds of servers, it is expensive, needs to be reconfigured when some of the servers are turned off during periods with low utilization and is not as robust as a distributed system. For this reason the use of multiple software based load balancers is preferential in a Cloud environment. While traditional load balancers often made use of the join-the-shortest-queue (JSQ) algorithm, as all the requests and responses tended to flow through the load balancer, a new class of distributed load balancers called Join-Idle-Queue (JIQ) for systems with multiple load balancers was introduced in [1]. Throughout this paper, which is an extended version[1] of [2], we use the terms load balancer and dispatcher interchangeably.

Distributed JIQ load balancing operates as follows: each dispatcher maintains an I-queue that contains a list of server identities. These are servers that reported that they became idle some time ago. We refer to these server identities as tokens. When a new job arrives at a dispatcher, it is immediately assigned to a server in the following manner:

- If the I-queue of the dispatcher is not empty, the job is assigned to a server, the identity of which is selected from the list in its I-queue. In such case the identity/token of the selected server is removed from the list.
- If the I-queue of the dispatcher is empty, a random server is selected.

From the server side we have that whenever a server becomes idle, it adds its identity to the I-queue of a dispatcher. Two algorithms are considered in [1]:

- JIQ-Random, meaning the dispatcher is selected at random,
- JIQ-SQ(d), meaning the server selects d dispatchers at random and adds its identity to a dispatcher with the shortest I-queue among the d selected dispatchers. For $d = 1$ this scheme coincides with JIQ-Random.

Notice that a server may hold one or several jobs even if it is listed in an I-queue of a dispatcher as other dispatchers may assign jobs to a server when their I-queue is empty. This can be avoided by demanding that an idle server *withdraws* its identity from the I-queue of the dispatcher as soon as a job is assigned by another dispatcher. As such we distinguish between the JIQ load balancing algorithm **with and without token withdrawals**. Without token withdrawals, one can make a further distinction on whether or not a server is allowed to add its token to more than one dispatcher as the server could become idle again before its outstanding token is used.

The performance of JIQ load balancing has been studied in the large-scale limit by various authors. When the number of dispatchers m remains fixed, while the number of servers n tends to infinity, JIQ is known to have vanishing waiting times in many settings [3, 4, 5] under subcritical load per server, that is, if λn denotes the total arrival rate and the mean job size equals 1, then all jobs are assigned to idle servers in the limit when $\lambda < 1$.

[1]Sections 6, 7 and 8 as well as Figures 1 and 2 are new.

As any load balancer can only support a finite number of servers in any real system, it might be more appropriate to look at the limit if both the number of dispatchers m and the number of servers n tend to infinity, such that $r = n/m$ remains fixed. This limit was initially considered in [1] for JIQ-Random and JIQ-SQ(d) under two assumptions that only hold for JIQ *with* token withdrawals. However, as we explain in Section 2, the limit presented in [1] is inexact as the authors mix properties of JIQ with and without withdrawals when deriving their result.

A mean field model for JIQ-Random and JIQ-Pod *with* token withdrawals and exponential job sizes is presented in [6]. JIQ-Pod operates as JIQ-Random, except that the power-of-d-choices paradigm [7, 8] is used by the dispatcher when a job arrives at a dispatcher with an empty I-queue. While JIQ-Pod improves the performance of JIQ-Random under high loads, the downside is that some jobs are not assigned immediately. While no convergence proofs are presented in [6], simulation results suggest that the unique fixed point of the mean field model corresponds to the exact limit. The authors in [6] also illustrate using simulation that their model for JIQ-Random is more accurate than the model in [1] for finite n, but no explanation is provided.

A number of variants of JIQ, including JIQ-SQ(d), *without* token withdrawals and exponential job sizes are analyzed using mean field models in [9] for the case where servers to not add their token to more than one dispatcher. These models are significantly more complicated than the ones with token withdrawals and even proving the existence of a unique fixed point was left as an open problem, which is needed before one can even start thinking about convergence proofs. The author does present simulation results that suggest that the models provide exact results in the large-scale limit. The main insights of the models in [9] are that the system behavior of JIQ-SQ(d) without token withdrawals is quite complex as the servers experience queue length dependent arrival rates and the order in which dispatchers select tokens from their I-queue impacts performance.

In this paper we make the following contributions:

1. We explain why the large-scale analysis presented in [1] is inexact. On the upside we show that the inaccuracy of the proposed limit for JIQ-SQ(d) is small and decreases rapidly as d increases, except loads close to one.
2. We introduce a novel mean field model for JIQ-SQ(d) with token withdrawals. We first consider exponential job sizes and then generalize to phase-type distributed job sizes. Both models are validated by simulation. Our results for JIQ-SQ(d) for exponential job sizes with $d = 1$ coincide with the results presented in [6] for JIQ-Random.
3. We generalize some of the mean field models in [9] to the setting with phase-type distributed job sizes and develop fast numerical methods to compute a fixed point. These models are also validated using simulation.
4. We compare the performance of JIQ load balancing with and without withdrawals using our mean field models and demonstrate the impact of the token selection method used by the dispatcher in case the tokens are

not withdrawn.

It should be relatively easy to prove that our mean field models become exact over finite time scales as the number of servers tends to infinity as our models fall within the framework of density dependent population processes of Kurtz [10].

Under the assumption (supported by simulation in Section 5 and 7) that the mean field models presented in this paper are asymptotically exact, the following insights are obtained for JIQ-SQ(d) with token withdrawals:

1. As n tends to infinity with $r = n/m$ fixed, the response time distribution of a job becomes identical to that in an M/PH/1 queue with load λq_0.
2. The value of q_0 depends on λ, d and r, but is independent of the job size distribution (with mean 1).
3. The method used by the dispatcher to select a token from its I-queue when an arrival occurs, has no impact on the performance.

When the tokens are not withdrawn, the large-scale queueing dynamics at the server side become more involved and the token selection method used by the dispatcher does affect performance. When comparing both variants using our mean field models, we observe that withdrawing tokens gives slightly better performance at low to medium loads (at the expense of adding work on the critical path), but at high loads token withdrawals have a negative impact on performance. Regarding the token selection method used by the dispatcher, we show that Last-Come-First-Served (LCFS) reduces the mean response time by a small margin compared to First-Come-First-Served (FCFS), which was also observed in [9] for $d = 1$ and exponential job sizes.

We end this introduction with a short discussion of some other JIQ related work. In [11] it was shown that JIQ is not heavy traffic optimal, which is not surprising due to the random assignments used when a job arrives at a dispatcher with an empty I-queue. The authors therefore propose and study Join-Below-Threshold load balancing which is in the same spirit as JIQ-Threshold [9]. Load balancers for homogeneous and heterogeneous systems using outdated queue length information were considered both in the case of a single [12] or multiple dispatchers [13, 14]. For heterogeneous systems that use JIQ load balancing with a fixed number of dispatchers, vanishing waiting times were achieved in the limit by exchanging tokens or using non-uniform token allotment in [15]. Finally, JIQ was also studied in a setting with service elasticity in [16] and [17].

The paper starts with a discussion of the large-scale limit analysis in [1] in Section 2. Sections 3 to 5 are devoted to the JIQ-SQ(d) models with withdrawals and their validation. Sections 6 and 7 discuss the mean field models for JIQ-SQ(d) without withdrawals, explain how to quickly determine a fixed point and validate the models using simulation. A performance comparison between the different JIQ-SQ(d) variants considered in this paper is presented in Section 8. The paper ends with some concluding remarks in Section 9.

n	q_0	$E[R]$
50	0.3738	1.4878
500	0.3757	1.4708
5000	0.3752	1.4687
model in [1]	0.4000	1.5152

Table 1: Simulation results for JIQ-Random with token withdrawals and exponential job sizes, $\lambda = 0.85$ and $r = n/m = 10$ for $n = 50, 500$ and 5000 servers. $E[R]$ is the mean response time and q_0 the probability that a dispatcher holds zero tokens. There is no convergence to the model as n tends to infinity.

2. Inexactness of an existing large-scale analysis

In this section we first demonstrate that the model in [1] for the large-scale system is inexact and identify the reason for this inexactness. In Table 1 we present an arbitrary simulation experiment for JIQ-Random with an increasing number of servers n and compare these results with the model in [1]. This model depends on two assumptions: (1) there is exactly one copy of each idle server in the I-queues and (2) there are only idle servers in the I-queues. As these assumptions are valid for JIQ-Random with token withdrawals, we simulated the system with token withdrawals. The fact that the model in [1] is not asymptotically exact for the system without token withdrawals was already demonstrated in [9, see Table I], which is not surprising as assumption (2) does not hold without withdrawals. Table 1 strongly suggests that the model in [1] is not asymptotically exact even with token withdrawals. Nevertheless, we will demonstrate that it may still be regarded as a good to excellent approximation.

To understand the cause of the inexactness, we look at Theorem 1 in [1]. Denote $\hat{q}_1$ as the probability that an I-queue contains at least one token in equilibrium when the number of servers n tends to infinity. Theorem 1 in [1] then states that for JIQ-Random, we have

$$\frac{\hat{q}_1}{1-\hat{q}_1} = (1-\lambda)r,$$

and for JIQ-SQ(d), we have

$$\sum_{i\geq 1} \hat{q}_1^{(d^i-1)/(d-1)} = (1-\lambda)r.$$

We explain below that the left-hand sides are the mean I-queue lengths for JIQ-Random and JIQ-SQ(d) *without* token withdrawals, while the right-hand side is the mean queue length of an I-queue in a system *with* token withdrawals. Hence Theorem 1 mixes two different JIQ systems and the analysis in [1] is therefore asymptotically inexact for both systems.

The left-hand sides of the above equations can be understood by looking at the mean field models in [9]. More specifically, looking at the ODEs in [9, Section IV.C] implies that the distribution of the number of tokens at an

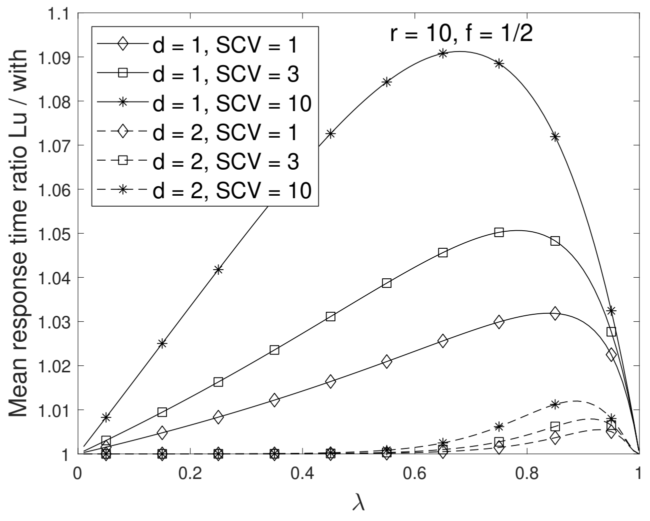

Figure 1: Impact of λ on mean response time ratio of the Lu approcimation and the mean field model for JIQ-SQ(d) **with withdrawals** presented in Section 4 for $d = 1$ and 2 and $SCV = 1, 3$ and 10.

I-queue for JIQ-Random is geometric with parameter $\hat{q}_1 = s_1/\lambda$, while for JIQ-SQ(d) the probability that an I-queue contains at least i tokens equals $\hat{q}_1^{(d^i-1)/(d-1)} = (s_1/\lambda)^{(d^i-1)/(d-1)}$. Therefore the average number of tokens in an I-queue is indeed given by $\frac{\hat{q}_1}{1-\hat{q}_1}$ for JIQ-Random and $\sum_{i\geq 1} \hat{q}_1^{(d^i-1)/(d-1)}$ for JIQ-SQ(d). The analysis in [9] is for the case *without* token withdrawals. Thus the expressions on the left-hand side in Theorem 1 in [1] are the mean I-queue lengths for the system *without* withdrawals. In such case the total number of tokens residing in the I-queues *does not* match the number of idle servers.

However the right-hand side equals $(1 - \lambda)r$ for both equations in Theorem 1 in [1] and this is the mean queue length of an I-queue in a system *with* token withdrawals. Indeed, when servers withdraw their token, the total number of tokens residing in the I-queues perfectly matches the number of idle servers. As $(1 - \lambda)$ should be the limiting fraction of idle servers and there are r times as many servers as dispatchers, the I-queue of a dispatcher contains on average $(1 - \lambda)r$ tokens.

In Figure 1 we present the ratio of the mean response time computed by the model in [1] with the mean response time of the mean field model presented in Section 4. Simulation experiments presented in Section 5 suggest that this model yields asymptotically exact results for JIQ-SQ(d) with token withdrawals. The jobs sizes with SCV equal to one correspond to exponential job sizes, the cases with $SCV > 1$ to an order 2 hyperexponential distribution with mean 1, $SCV = 3$ or 10 and $f = 1/2$, where the latter means that we have balanced means for both phases, that is, $p_1/\mu_1 = p_2/\mu_2$. Figure 1 confirms the asymptotic inexactness of the Lu model for $r = 10$. We also performed the same experiment

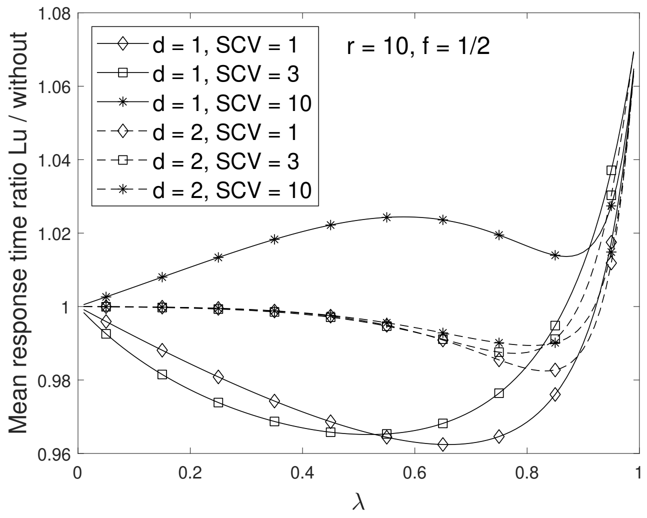

Figure 2: Impact of λ on mean response time ratio of the Lu approcimation and the mean field model for JIQ-SQ(d) **without withdrawals** and FCFS token selection presented in Section 6 for $d = 1$ and 2 and $SCV = 1, 3$ and 10.

for other r values and noted that the errors tend to decrease with increasing r. The figure also clearly shows that the accuracy improves as d increases from 1 to 2. Additional experiments (not presented) showed that the accuracy improved even further when considering larger d values. The figure does suggest that the Lu approximation yields asymptotically exact results when λ tends to one for JIQ-SQ(d) with withdrawals.

In Figure 2 we perform a similar experiment as in Figure 1, but now with JIQ-SQ(d) without withdrawals and FCFS token selection. The errors are all well below 10%, but the approximation is not exact for JIQ-SQ(d) without replacement when λ tends to one. It is fair to state that while the model in [1] is not asymptotically exact for JIQ-SQ(d) with or without withdrawals, its accuracy is outstanding for $d > 1$ whenever λ is not too close to one (and r is sufficiently large).

3. Mean Field Model for JIQ-SQ(d) with withdrawals and exponential job sizes

In this section we present a new mean field model for JIQ-SQ(d) with servers that withdraw their token when a job is assigned by another dispatcher. We consider a system with m dispatchers, that each have an I-queue to hold tokens, and n servers. As before let $r = n/m$. Poisson arrivals occur at rate λn and are spread uniformly over the m dispatchers. For now the job size is exponentially distributed with mean 1.

Let $Q_i(t)$ be the number of I-queues with exactly i tokens at time t, $S_i(t)$ the number of servers with i jobs at time t. Hence, $\sum_i Q_i(t) = m$ and $\sum_i S_i(t) = n$.

Let $\hat{Q}_k(t) = \sum_{i\geq k} Q_i(t)$ be the number of I-queues holding k or more tokens. Define the fractions $q_i(t) = Q_i(t)/m, \hat{q}_i(t) = \hat{Q}_i(t)/m$ and $s_i(t) = S_i(t)/n$. Let $\Delta\hat{Q}_i(t)$ denote the expected change in $\hat{Q}_i(t)$ over a small interval dt, that is, $\Delta\hat{Q}_i(t) = E[\hat{Q}_i(t+dt) - \hat{Q}_i(t)]$. As explained below, we have for $i > 0$

$$\begin{aligned}\Delta\hat{Q}_i(t) = -(\lambda n)dt\left(\frac{\hat{Q}_i(t)}{m} - \frac{\hat{Q}_{i+1}(t))}{m}\right) + S_1(t)dt\left(\frac{\hat{Q}_{i-1}(t)^d}{m^d} - \frac{\hat{Q}_i(t)^d}{m^d}\right)\\ - (\lambda n)dtq_0(t)i\left(\frac{\hat{Q}_i(t)}{n} - \frac{\hat{Q}_{i+1}(t)}{n}\right).\end{aligned}$$

The first term is due to arrivals were $(\hat{Q}_i(t) - \hat{Q}_{i+1}(t))/m$ is the probability that a random arrival occurs at a dispatcher with exactly i tokens in its I-queue. The second term is due to service completions as jobs are assumed to be exponential in size with mean one and a server that becomes idle uses the power-of-d-choices rule to select a dispatcher. The last term is due to servers withdrawing tokens and can be understood as follows. First note that $i(\hat{Q}_i(t) - \hat{Q}_{i+1}(t))$ is the total number of tokens residing at the I-queues that hold exactly i tokens. Therefore $i(\hat{Q}_i(t) - \hat{Q}_{i+1}(t))/n$ is the probability that an arrival at an empty I-queue is assigned to a server that has a token at an I-queue with length i.

Dividing left and right by mdt yields

$$\begin{aligned}\frac{d\hat{q}_i(t)}{dt} &= -\lambda r(\hat{q}_i(t) - \hat{q}_{i+1}(t)) + s_1(t)r(\hat{q}_{i-1}(t)^d - \hat{q}_i(t)^d)\\ &\quad - \lambda(1-\hat{q}_1(t))i(\hat{q}_i(t) - \hat{q}_{i+1}(t)),\end{aligned} \tag{1}$$

where $\hat{q}_0(t) = 1$ for all t.

We now proceed with the servers for $i > 0$:

$$\begin{aligned}\Delta S_i(t) = (\lambda n)dtq_0(t)\left(\frac{S_{i-1}(t)}{n} - \frac{S_i(t))}{n}\right) - dt\left(S_i(t) - S_{i+1}(t)\right)\\ + 1[i=1](\lambda n)(1-q_0(t))dt.\end{aligned}$$

The first term corresponds to arrivals in empty I-queues, the second to service completions and the last term to arrivals in a non-empty I-queue. Dividing by ndt yields

$$\frac{ds_i(t)}{dt} = \lambda q_0(t)(s_{i-1}(t) - s_i(t)) - (s_i(t) - s_{i+1}(t)) + 1[i=1]\lambda(1-q_0(t)). \tag{2}$$

Similarly, we find

$$\frac{ds_0(t)}{dt} = -\lambda q_0(t)s_0(t) + s_1(t) - \lambda(1-q_0(t)). \tag{3}$$

Notice that all the above equations apply irrespective of the manner in which a dispatcher selects a token from its I-queue, which is in contrast to JIQ-SQ(d) without token withdrawals [9].

Theorem 1. *Let $(s, \hat{q})$ be a fixed point of* (1)-(3) *such that $\sum_{i\geq 0} s_i = 1$, $\sum_{i\geq 0} i s_i < \infty$, $\hat{q}_0 = 1$ and $\sum_{i\geq 0} \hat{q}_i < \infty$, then $\lambda = \sum_{i\geq 1} s_i$,*

$$s_1 = \lambda(1 - \lambda q_0), \tag{4}$$

$$s_k = s_1(\lambda q_0)^{k-1}, \tag{5}$$

for $k > 1$, with $q_0 = 1 - \hat{q}_1$. Further,

$$\sum_{i\geq 1} \hat{q}_i = (1 - \lambda)r. \tag{6}$$

Proof. The equality $\lambda = \sum_{i\geq 1} s_i$ follows from (2) as

$$0 = \sum_{i\geq 1} i \frac{ds_i(t)}{dt} = \lambda q_0(t) \sum_{i\geq 0} s_i(t) - \sum_{i\geq 1} s_i(t) + \lambda(1 - q_0(t)).$$

Using (2) and the fact that $\sum_{i\geq k} \frac{ds_i(t)}{dt} = 0$ yields

$$s_k = s_{k-1}\lambda q_0,$$

for $k \geq 2$, which implies (5). The expression for s_1 in (4) is now immediate by combining $\lambda = \sum_{i\geq 1} s_i$ with (5). To prove (6) we note that (1) implies

$$0 = \sum_{i\geq 1} \frac{d\hat{q}_i(t)}{dt} = -\hat{q}_1(t)\lambda r + s_1(t) r - \lambda(1 - \hat{q}_1(t)) \sum_{i\geq 1} \hat{q}_i(t).$$

When combined with (4), we find that $\sum_{i\geq 1} \hat{q}_i = (1 - \lambda)r$. $\square$

Remarks: 1) Looking at the expression for s_k in (5), we see that the queue length distribution is identical to an M/M/1 queue with arrival rate λq_0 when the server is busy and with an increased arrival rate when the queue is idle (such that the probability that the queue is idle is $1 - \lambda$ instead of $1 - \lambda q_0$). As increasing the arrival rate in an idle queue does not impact the response time distribution, jobs have the same response time distribution as in an ordinary M/M/1 queue with arrival rate λq_0. Therefore the response time is exponential with parameter $1 - \lambda q_0$ (as the service rate equals 1). We indicate how to compute q_0 further on.

2) The equality $\lambda = \sum_{i\geq 1} s_i$ is natural as λ should be the probability that a server is busy. The equality in (6) is also expected as every idle server has exactly one token at one of the dispatchers and the fraction of idle servers is $(1 - \lambda)$ while there are r times as many servers as dispatchers. Therefore the mean number of tokens per dispatcher should be $(1 - \lambda)r$.

Theorem 2. *Let $(s, \hat{q})$ be a fixed point of* (1)-(3) *such that $\sum_{i\geq 0} s_i = 1$, $\sum_{i\geq 0} i s_i < \infty$, $\hat{q}_0 = 1$ and $\sum_{i\geq 0} \hat{q}_i < \infty$, then*

$$\hat{q}_{k+1} = \hat{q}_k - (\hat{q}_{k-1}^d - \hat{q}_k^d) \frac{1 - \lambda q_0}{1 + \frac{k q_0}{r}}, \tag{7}$$

for $k \geq 1$ with $q_0 = 1 - \hat{q}_1$.

Proof. For any fixed point we have $\sum_{i\geq k} \frac{d\hat{q}_i(t)}{dt} = 0$, which implies that

$$0 = -\hat{q}_k\lambda r + s_1 r\hat{q}_{k-1}^d - \lambda(1-\hat{q}_1)\left(k\hat{q}_k + \sum_{s>k}\hat{q}_s\right),$$

for $k \geq 1$. We can rewrite this as

$$\hat{q}_k = \frac{s_1 r\hat{q}_{k-1}^d - \lambda q_0\left((1-\lambda)r - \sum_{s=1}^{k-1}\hat{q}_s\right)}{\lambda r + \lambda q_0(k-1)}, \tag{8}$$

due to (6). Although this expression can be used to compute $\hat{q}_k$, we derive a more elegant recursion that is equivalent. We first note that by (8)

$$\hat{q}_k(\lambda r + \lambda q_0 k) = \left(s_1 r\hat{q}_{k-1}^d - \lambda q_0\left((1-\lambda)r - \sum_{s=1}^{k-1}\hat{q}_s\right)\right)\left(1 + \frac{q_0}{r+q_0(k-1)}\right),$$

and

$$\hat{q}_{k+1}(\lambda r + \lambda q_0 k) = s_1 r\hat{q}_k^d - \lambda q_0\left((1-\lambda)r - \sum_{s=1}^{k}\hat{q}_s\right).$$

Hence,

$$\begin{aligned}(\hat{q}_k - \hat{q}_{k+1})(\lambda r + \lambda q_0 k) &= s_1 r(\hat{q}_{k-1}^d - \hat{q}_k^d) - \lambda q_0\hat{q}_k + \\ &\left(s_1 r\hat{q}_{k-1}^d - \lambda q_0\left((1-\lambda)r - \sum_{s=1}^{k-1}\hat{q}_s\right)\right)\frac{q_0}{r+q_0(k-1)} \\ &= s_1 r(\hat{q}_{k-1}^d - \hat{q}_k^d) - \lambda q_0\hat{q}_k + \lambda q_0\hat{q}_k.\end{aligned}$$

When combined with (4) this proves (7). □

Remarks: 1) When $d = 1$, (7) implies that

$$q_k = q_0\prod_{\ell=1}^{k}\frac{1-\lambda q_0}{1+\frac{\ell q_0}{r}},$$

where $q_k = \hat{q}_k - \hat{q}_{k+1}$. Further, as $\sum_{i\geq 0} q_i = \hat{q}_0 = 1$ this shows that q_0 is a solution of

$$q_0\sum_{k\geq 0}\prod_{\ell=1}^{k}\frac{1-\lambda q_0}{1+\frac{\ell q_0}{r}} = 1,$$

which coincides with Theorem 2 in [6], where it is shown that q_0 is the unique solution of this equation on $(0,1)$.
2) Given q_0 we can compute $\hat{q}_k$ for $k \geq 2$ using (7) (or (8)) as $\hat{q}_1 = 1 - q_0$.

Theorem 3. *There exists a unique fixed point $(s, \hat{q})$ of (1)-(3) such that $\sum_{i\geq 0} s_i = 1$, $\sum_{i\geq 0} i s_i < \infty$, $\hat{q}_0 = 1$ and $\sum_{i\geq 0}\hat{q}_i < \infty$.*

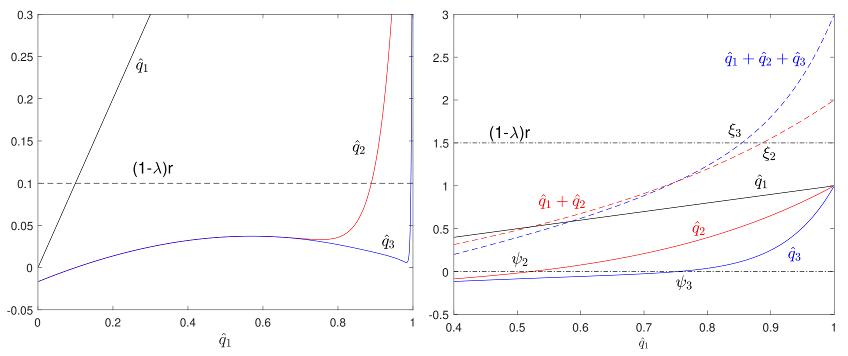

Figure 3: Illustration of non-monotone behavior for $\lambda = 0.98$, $r = 5$ and $d = 20$ (top), ξ_k and ψ_k for $\lambda = 0.7$, $r = 5$, and $d = 3$ (bottom).

Proof. Given Theorems 1 and 2, it suffices to show that there is a unique $\hat{q}_1 \in (0, 1)$ such that (6) holds, that is, such that $\sum_{i\geq 1} \hat{q}_i = (1-\lambda)r$. It is clear from (7) that $\hat{q}_k$ is a continuous function of $\hat{q}_1$ (but not necessarily monotone, see Figure 3(left) for an example) and when $\hat{q}_1 = 1$, then $\hat{q}_k = 1$ for any k. Therefore for any $k > (1-\lambda)r$ we have for $\hat{q}_1$ sufficiently close to 1 that $\sum_{s=1}^{k} \hat{q}_s > (1-\lambda)r$. Further $\hat{q}_{k+1} \leq \hat{q}_k$, meaning $\hat{q}_k \leq 0$ for any k when $\hat{q}_1 = 0$. Define $\xi_k = 1$ for $k = 1, \ldots, \lfloor (1-\lambda)r \rfloor$ as the smallest value of $\hat{q}_1 \in (0, 1)$ such that $\sum_{s=1}^{k} \hat{q}_s = (1-\lambda)r$ for $k > \lfloor (1-\lambda)r \rfloor$.

We now argue that the following two statements hold:

1. $\hat{q}_k > 0$ when $\sum_{s=1}^{k-1} \hat{q}_s \geq (1-\lambda)r$,
2. $\hat{q}_k$ has a positive derivative on $(0, \xi_{k-1}]$.

The first statement is immediate from (8) as both $s_1 r \hat{q}_{k-1}^d$ and $\lambda r + \lambda q_0(k-1)$ are positive for any $\hat{q}_1 \in (0, 1)$. Therefore $\sum_{s=1}^{k} \hat{q}_s > (1-\lambda)r$ when $\hat{q}_1$ equals ξ_{k-1}, which shows that $\xi_k \leq \xi_{k-1}$. In other words, $\xi_k \in (0, 1]$ is non-increasing in k as illustrated in Figure 3(right).

The second statement follows by induction on k as follows. The statement clearly holds for $k = 1$. For $k > 1$, we have by induction that that $\hat{q}_s$ is increasing on $(0, \xi_{s-1})$ for $s < k$, meaning $\sum_{s=1}^{k-1} \hat{q}_s$ is increasing on $(0, \xi_{k-1})$ as ξ_s is non-increasing in s. By definition of ξ_{k-1}, we also have that $\sum_{s=1}^{k-1} \hat{q}_s < (1-\lambda)r$ on $(0, \xi_{k-1})$. By (8) we now see that $\hat{q}_k$ is increasing on $(0, \xi_{k-1})$ as $s_1 r \hat{q}_{k-1}^d$ is increasing and positive on $(0, \xi_{k-1})$, $\lambda r + q_0(k-1)/r$ is decreasing and positive on $(0, \xi_{k-1})$ and $-\lambda q_0((1-\lambda)r - \sum_{s=1}^{k-1} \hat{q}_s)$ is negative and increasing on $(0, \xi_{k-1})$.

We proceed by using induction on k to argue that $\sum_{s=1}^{k} \hat{q}_s > (1-\lambda)r$ when $\hat{q}_1 \in (\xi_k, 1)$. This trivially holds for $k = 1$. For $k > 1$, we have by induction that $\sum_{s=1}^{k-1} \hat{q}_s$ exceeds $(1-\lambda)r$ on $(\xi_{k-1}, 1)$. By (8), we therefore have that $\hat{q}_k$ is positive on $(\xi_{k-1}, 1)$, so $\sum_{s=1}^{k} \hat{q}_s > \sum_{s=1}^{k-1} \hat{q}_s > (1-\lambda)r$. For $\hat{q}_1 \in (\xi_k, \xi_{k-1})$, we know that $\hat{q}_k$ is increasing and $\sum_{s=1}^{k} \hat{q}_s = (1-\lambda)r$ for $\hat{q}_1 = \xi_k$. This shows that for $k > \lfloor (1-\lambda)r \rfloor$, there is a unique solution $\hat{q}_1 = \xi_k$ on $(0, 1)$ such that $\sum_{s=1}^{k-1} \hat{q}_s = (1-\lambda)r$.

The unique value of q_0 such that $\sum_{i\geq 1} \hat{q}_i = (1-\lambda)r$ is then found as $1 - \lim_{k\to\infty} \xi_k$ as $q_0 = 1 - \hat{q}_1$ and the limit of a decreasing sequence of values in $(0,1)$ exists. □

Remarks: 1) We see from the proof of Theorem 3 that for any $k > 1$ there exists a unique ψ_k such that $\hat{q}_k = 0$ if $\hat{q}_1 = \psi_k$. Further $\psi_k \geq \psi_{k-1}$ as $\hat{q}_k \leq \hat{q}_{k-1}$ (see Figure 3(right) for an illustration). This implies that if $\hat{q}_1 < \lim_{k\to\infty} \xi_k$, then $\lim_{k\to\infty} \hat{q}_k < 0$, while $\lim_{k\to\infty} \hat{q}_k > 0$ if $\hat{q}_1 > \lim_{k\to\infty} \xi_k$. Hence, $\lim_{k\to\infty} \xi_k = \lim_{k\to\infty} \psi_k$, where ξ_k is a decreasing sequence and ψ_k an increasing one.

2) Looking at the proof of Theorem 3 and the previous remark, we have the following simple algorithm to compute $q_0 = 1 - \hat{q}_1$. We start with $k = \lceil (1-\lambda)r \rceil$ and determine ξ_k and ψ_k by performing a bisection algorithm on $(0,1)$ using (7). Next we repeatedly increase k by one and determine ξ_k and ψ_k using a bisection algorithm on (ψ_{k-1}, ξ_{k-1}) until $\xi_k - \psi_k < 10^{-15}$ (see Algorithm 1). Once q_0 is found, we can compute the queue length distributions s and $\hat{q}$ using (4), (5) and (7). As noted before, the response time distribution of a job is exponential with parameter $1 - \lambda q_0$.

```
Input: $\lambda$, $d$, $r$
Output: $q_0$
1 $k := \lceil (1-\lambda)r \rceil$;
2 Compute $\xi_k$ using bisection on $(0,1)$;
3 Compute $\psi_k$ using bisection on $(0,\xi_k)$;
4 while $\xi_k - \psi_k > 10^{-15}$ do
5     $k := k+1$;
6     Compute $\xi_k$ using bisection on $(\psi_{k-1}, \xi_{k-1})$;
7     Compute $\psi_k$ using bisection on $(\psi_{k-1}, \xi_k)$;
8 end
9 $q_0 := 1 - \xi_k$;
```

Algorithm 1: Computation of q_0 for JIQ-SQ(d).

4. Mean Field Model for JIQ-SQ(d) with withdrawals and phase-type job sizes

We now generalize the previous mean field model to phase-type job sizes characterized by a $1 \times b$ vector α and a $b \times b$ matrix T such that $P[X > t] = \alpha \exp(Tt)e$, where X is the job size and e a vector of ones. For further use denote $t^* = (-T)e$. Phase-type (PH) distributions are distributions with a modulating finite state Markov chain (see also [18]). Any general positive-valued distribution can be approximated arbitrarily close with a PH distribution and there are various fitting tools available for PH distributions (see e.g. [19]). The main objective of this section is to show the following two properties:

1) The distribution of the number of tokens in an I-queue (given by $\hat{q}$) does not depend on the job size distribution. Hence, we have insensitivity in the

mean field model for the manner in which the tokens are distributed over the dispatchers. This also implies that the probability q_0 that an I-queue is empty is insensitive to the job size distribution.

2) The response time distribution of a job under JIQ-SQ(d) with phase-type job size distribution X is identical to that of a job arriving in an M/G/1 queue with arrival rate λq_0 and job size distribution X. Note that this was already shown for exponential job sizes in the previous section.

Let $S_{i,j}(t)$ be the number of servers in service phase j with i jobs at time t and $S_0(t)$ be the number of idle servers at time t. Denote $\vec{S}_i(t) = (S_{i,1}(t), \ldots, S_{i,b}(t))$, $s_{i,j}(t) = S_{i,j}(t)/n$ and $\vec{s}_i(t) = \vec{S}_i(t)/n$. If we look at the evolution of the number of tokens in an I-queue we see that it is identical to the case of exponential job sizes, except for the term that corresponds to the service completions. In the exponential case this term was given by

$$S_1(t)dt\left(\frac{\hat{Q}_{i-1}(t)^d}{m^d} - \frac{\hat{Q}_i(t)^d}{m^d}\right),$$

while in case of phase-type job sizes this becomes

$$\left(\sum_j S_{1,j}(t)t_j^*\right)dt\left(\frac{\hat{Q}_{i-1}(t)^d}{m^d} - \frac{\hat{Q}_i(t)^d}{m^d}\right),$$

where the first sum can be written in matrix notation as $\vec{s}_1(t)t^*$. This implies that (1) becomes

$$\begin{aligned}\frac{d\hat{q}_i(t)}{dt} &= -\lambda r(\hat{q}_i(t) - \hat{q}_{i+1}(t)) + (\vec{s}_1(t)t^*)r(\hat{q}_{i-1}(t)^d - \hat{q}_i(t)^d)\\ &\qquad - \lambda(1-\hat{q}_1(t))i(\hat{q}_i(t) - \hat{q}_{i+1}(t)),\end{aligned} \tag{9}$$

which is identical to (1) if we replace $s_1(t)$ by $\vec{s}_1(t)t^*$.

We now proceed with the expected change in $S_{i,j}(t)$:

$$\begin{aligned}\Delta S_{i,j}(t) = (\lambda n)dtq_0(t)&\left(1[i>1]\frac{S_{i-1,j}(t)}{n} + 1[i=1]\frac{S_0(t)}{n}\alpha_j - \frac{S_{i,j}(t))}{n}\right)\\ &- dtS_{i,j}(t)t_j^* + dt\sum_k S_{i+1,k}(t)t_k^*\alpha_j - dtS_{i,j}(t)\sum_{k\neq j}T_{j,k} + dt\sum_{k\neq j}S_{i,k}(t)T_{k,j}\\ &+ 1[i=1](\lambda n)(1-q_0(t))\alpha_j dt.\end{aligned}$$

The first and last term are very similar to (2), the other terms correspond to service completions and phase changes. As $t^* = -Te$, this can be written in matrix form as

$$\begin{aligned}\Delta\vec{S}_i(t) &= (\lambda n)dtq_0(t)\left(1[i>1]\frac{S_{i-1,j}(t)}{n}+1[i=1]\frac{S_0(t)}{n}\alpha-\frac{\vec{S}_i(t))}{n}\right)\\ &\quad - dt\vec{S}_i(t)diag(t^*)+dt\vec{S}_{i+1}(t)t^*\alpha\\ &\quad + dt\vec{S}_i(t)(diag(t^*)+diag(T))+dt(\vec{S}_i(t)T\\ &\quad - \vec{S}_i(t)diag(T))+1[i=1](\lambda n)(1-q_0(t))\alpha dt,\end{aligned}$$

where $diag(v)$ of a vector v is a diagonal matrix with v on its main diagonal and $diag(A)$ of a matrix A is the diagonal matrix obtained by setting all off diagonal entries of A to zero. After simplifying this implies that

$$\begin{aligned}\frac{d\vec{s}_i(t)}{dt} &= \lambda q_0(t)(1[i>1]\vec{s}_{i-1}(t)+1[i=1]s_0(t)\alpha-\vec{s}_i(t))\\ &\quad+\vec{s}_{i+1}(t)t^*\alpha+\vec{s}_i(t)T+1[i=1]\lambda(1-q_0(t))\alpha. \qquad (10)\end{aligned}$$

Similarly, we find

$$\frac{ds_0(t)}{dt} = -\lambda q_0(t)s_0(t)+\vec{s}_1(t)t^*-\lambda(1-q_0(t)). \qquad (11)$$

Theorem 4. *Let $(s,\hat{q})$ be a fixed point of (9-11) such that $\sum_{i\geq 0}\vec{s}_i e = 1$, $\sum_{i\geq 0} i\vec{s}_i e<\infty$, $\hat{q}_0=1$ and $\sum_{i\geq 1}\hat{q}_i<\infty$ then $\lambda=\sum_{i\geq 1}\vec{s}_i e$,*

$$\vec{s}_1 = (1-\lambda q_0)\alpha R/q_0, \qquad (12)$$

$$\vec{s}_k = \vec{s}_1 R^{k-1}, \qquad (13)$$

for $k>1$, with $q_0=1-\hat{q}_1$ and $R=\lambda q_0(\lambda q_0 I-\lambda q_0 e\alpha - T)^{-1}$. Further, $\vec{s}_1 t^* = \lambda(1-\lambda q_0)$.

Proof. By demanding that $\sum_{i\geq 1}\frac{d\vec{s}_i(t)}{dt}+s_0(t)\alpha=0$ and using the definition of t^*, one finds that

$$\sum_{i\geq 1}\vec{s}_i(T+t^*\alpha)=0.$$

Hence, $\sum_{i\geq 1}\vec{s}_i$ is a multiple of the unique invariant vector β for which $\beta(T+t^*\alpha)=0$ and $\beta e=1$ holds. By considering the equality $\sum_{i\geq 1} i\frac{d\vec{s}_i(t)}{dt}e=0$, one also finds that

$$\sum_{i\geq 1}\vec{s}_i t^* = \lambda.$$

As $\beta t^*=1$, this allows us to conclude that $\sum_{i\geq 1}\vec{s}_i=\lambda\beta$ and thus that $s_0=1-\lambda$. We may therefore write

$$\lambda q_0 s_0+\lambda(1-q_0)=\lambda\frac{1-\lambda q_0}{1-\lambda}s_0,$$

which means that the fixed point equations associated with (10) and (11) can be stated as

$$0 = 1[i > 1]\lambda q_0 \vec{s}_{i-1} - \lambda q_0 \vec{s}_i + 1[i = 1]\lambda \frac{1 - \lambda q_0}{1 - \lambda} s_0 \alpha + \vec{s}_{i+1} t^* \alpha + \vec{s}_i T,$$

and

$$0 = -\lambda \frac{1 - \lambda q_0}{1 - \lambda} s_0 + \vec{s}_1 t^*.$$

These fixed point equations are identical to the balance equations of an M/PH/1 queue with arrival rate $\lambda_0 = \lambda(1 - \lambda q_0)/(1 - \lambda)$ when the queue is empty and arrival rate λq_0 otherwise. Therefore (12) and (13) hold due to Theorem 2 in [20]. To verify that $\vec{s}_1 t^* = \lambda(1 - \lambda q_0)$, we note that

$$(\lambda q_0 I - \lambda q_0 e\alpha - T)e = t^*,$$

which shows that $Rt^* = \lambda q_0 e$ and therefore by (12) we find $\vec{s}_1 t^* = \lambda(1 - \lambda q_0)$ as required. □

Theorem 5. *There exists a unique fixed point* $(\vec{s}, \hat{q})$ *of* (9)-(11) *such that* $\sum_{i\geq 0} s_i = 1$, $\sum_{i\geq 0} i s_i < \infty$, $\hat{q}_0 = 1$ *and* $\sum_{i\geq 0} \hat{q}_i < \infty$. *Further,* $\hat{q}$ *is insensitive to the phase-type distribution* (α, T).

Proof. By Theorem 4 we have that $\vec{s}_1 t = \lambda(1 - q_0 \lambda)$, therefore the fixed point equations associated with (1) and (9) are identical and have a unique solution $\hat{q}$ due to Theorem 3. As $q_0 = 1 - \hat{q}_1$, the $\vec{s}$ part of the fixed point $(\vec{s}, \hat{q})$ is fully determined by (12) and (13). □

Remarks: 1) The unique fixed point $(\vec{s}, \hat{q})$ can be computed by first computing q_0 in the same manner as in the exponential case. The vectors $\vec{s}_k$, for $k \geq 1$ are then computed using (12) and (13).

2) The fixed point $(\vec{s}, \hat{q})$ is such that the distribution $\vec{s}$ at the servers is identical to an M/PH/1 queue with arrival rate λq_0 and an increased arrival rate λ_0 when the queue is empty (such that the probability to have an empty queue is $1 - \lambda$). As such the response time distribution is the same as in an M/G/1-queue with arrival rate λq_0, therefore the mean response time $E[R]$ is given by the Pollaczek–Khinchin mean value formula:

$$E[R] = \frac{\lambda q_0 E[X^2]}{2(1 - \lambda q_0)} + 1,$$

as the mean job size $E[X] = 1$.

5. Validation of the JIQ-SQ(d) with withdrawals mean field model

In this section we present simulation results to verify the accuracy of the mean field models presented in Sections 3 and 4. We performed a large number

settings	Job sizes	n	q_0	$E[R]$	rel.err. %
$\lambda = 0.7$ $r = 5$ $d = 3$	Exponential	50	0.1682	1.1457	0.0341
		500	0.1482	1.1173	0.0030
		5000	0.1463	1.1141	0.0003
		∞	0.1460	1.1138	
	HypExp $SCV = 10$ $f = 1/2$	50	0.1712	1.8476	0.1364
		500	0.1477	1.6428	0.0104
		5000	0.1458	1.6271	0.0007
		∞	0.1460	1.6259	
$\lambda = 0.85$ $r = 10$ $d = 1$	Exponential	50	0.3738	1.4878	0.0132
		500	0.3757	1.4708	0.0017
		5000	0.3752	1.4687	0.0002
		∞	0.3753	1.4684	
	HypExp $SCV = 10$ $f = 1/10$	50	0.3790	3.8480	0.0707
		500	0.3765	3.6012	0.0070
		5000	0.3757	3.5825	0.0018
		∞	0.3753	3.5761	

Table 2: JIQ-SQ(d) with replacement: approximation error of the simulated q_0 and $E[R]$ and relative error of $E[R]$.

of simulation experiments and present some arbitrarily selected cases in Table 2. We performed experiments with increasing values for the number of servers $n = 50, 500$ and 5000. The number of dispatchers used equals $m = n/r$. The system was simulated for a length of $10^7/m$ time units (where the mean job size equals one time unit). A warm-up period of 33% was used and the results were averaged over several runs. Apart from the simulation results, Table 2 also contains the value of q_0 determined by Algorithm 1 and the mean response time $E[R]$ of the corresponding M/PH/1 queue with load λq_0. They are presented on the line with $n = \infty$.

We considered both exponential job sizes and hyper-exponential job sizes with mean 1. In the latter case, we set the parameters p, μ_1, μ_2, with $\alpha = (p, 1-p)$ and $T = diag(-\mu_1, -\mu_2)$, such that the squared coefficient of variation (SCV) equals 10 and $f = p/\mu_1$. Table 2 clearly suggests that the mean field models of Sections 3 and 4 are asymptotically exact as the simulation results appear to converge towards the performance predicted by the unique fixed point of the corresponding mean field model.

Having validated the mean field models, it seems fair to state that for JIQ-SQ(d) with token withdrawals the response time of a job approaches the response time in an ordinary M/PH/1 queue with load λq_0 as the number of servers n becomes large (with $r = n/m$ fixed). Hence, the system behavior on the server side becomes very simple. Unfortunately determining q_0 is much harder due to the withdrawal of tokens. If we consider JIQ-SQ(d) without token withdrawals, then [9] suggests that the dynamics at the dispatcher side become very simple, e.g., for $d = 1$ the number of tokens has a geometric distribution. However,

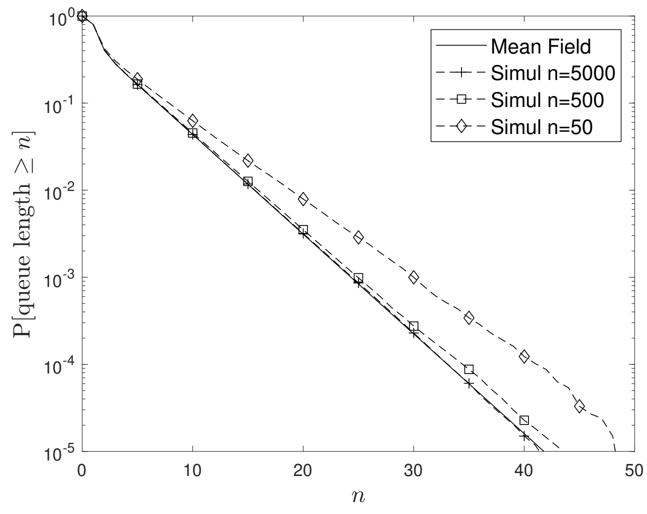

Figure 4: Validation of the queue length distribution at server side for $\lambda = 0.8$, $d = 2$, $r = 5$ and hyper-exponential job sizes with SCV=10 and $f = 1/2$.

without token withdrawals the server dynamics are much more involved as the job arrival rates become queue length dependent. The mean field model in [1] can be regarded as combining the simple server side dynamics of JIQ-SQ(d) with token withdrawals with the simple dispatcher dynamics of JIQ-SQ(d) without token withdrawals. Remarkably, this combination yields a good to excellent approximation as shown in Figures 1 and 2.

We also performed several simulation experiments that demonstrated that our mean field models are also very accurate for the queue length distribution at the servers (and not just the mean), only one such example is presented in Figure 4.

6. Mean Field Model for JIQ-SQ(d) without withdrawals and phase-type job sizes

In this section we extend and combine some of the mean field models in [9] that considered exponential job sizes to a setting with phase-type distributed job sizes. Let $\hat{q}_i(t)$ be the fraction of dispatchers with i or more tokens at time t, for $i \geq 1$, $s_{0,j}$ the fraction of idle servers with a token in position j at time t for $j > 0$ and let entry k of the vector $\vec{s}_{i,j}(t)$, for $i > 0$ and $j \geq 0$, be the fraction of servers with i jobs, service phase k and a token in position j where $j = 0$ means that the server does not have a token at a dispatcher.

6.1. Drift equations with FCFS token selection

This model focuses on JIQ-SQ(d) without token withdrawals and considers FCFS token selection at the dispatcher. It generalizes the model in [9, Section

IV.C] to phase-type distributed job sizes. On the side of the dispatcher we have

$$\frac{d\hat{q}_i(t)}{dt} = -\lambda r(\hat{q}_i(t) - \hat{q}_{i+1}(t)) + \vec{s}_{1,0}(t)t^* r(\hat{q}_{i-1}(t)^d - \hat{q}_i(t)^d). \tag{14}$$

The first term corresponds to arrivals that remove a token from a dispatcher with exactly i tokens, while the second term corresponds to servers becoming idle that place a token at a dispatcher with exactly $i-1$ tokens. On the server side we find for $i > 1$ and $j \geq 1$

$$\begin{aligned}\frac{d\vec{s}_{i,j}(t)}{dt} &= \vec{s}_{i,j}(t)T + \vec{s}_{i+1,j}(t)t^*\alpha + \lambda q_0(t)(\vec{s}_{i-1,j}(t) - \vec{s}_{i,j}(t)) \\ &\quad + \lambda r(\vec{s}_{i,j+1}(t) - \vec{s}_{i,j}(t)).\end{aligned} \tag{15}$$

Here the first term captures both phase changes and service completions in a server with i jobs and a token in position j. The second corresponds to service completions in servers with $i+1$ jobs and a token in position j. The next term corresponds to jobs that are dispatched to a server because they arrived at a dispatcher without tokens and the final term is due to arrivals in the dispatcher holding the token of a server with i jobs. For $i = 1$ and $j \geq 1$ we get the same terms except for the third term where a new phase must be selected when a job is dispatched to an idle server:

$$\begin{aligned}\frac{d\vec{s}_{1,j}(t)}{dt} &= \vec{s}_{1,j}(t)T + \vec{s}_{2,j}(t)t^*\alpha + \lambda q_0(t)(s_{0,j}(t)\alpha - \vec{s}_{1,j}(t)) \\ &\quad + \lambda r(\vec{s}_{1,j+1}(t) - \vec{s}_{1,j}(t)).\end{aligned} \tag{16}$$

The drift for $i > 1$ and $j = 0$ is similar to the drift with $j > 0$, but now the fourth term changes as the server has no token when $j = 0$ and a job is assigned if the token is in position 1:

$$\frac{d\vec{s}_{i,0}(t)}{dt} = \vec{s}_{i,0}(t)T + \vec{s}_{i+1,0}(t)t^*\alpha + \lambda q_0(t)(\vec{s}_{i-1,0}(t) - \vec{s}_{i,0}(t)) + \lambda r\vec{s}_{i-1,1}(t). \tag{17}$$

Finally, for $i = 1$ the last two terms change as idle servers always have a token placed at some dispatcher and a new phase must be selected if a job is assigned to an idle server:

$$\frac{d\vec{s}_{1,0}(t)}{dt} = \vec{s}_{1,0}(t)T + \vec{s}_{2,0}(t)t^*\alpha - \lambda q_0(t)\vec{s}_{1,0}(t) + \lambda r s_{0,1}(t)\alpha. \tag{18}$$

For the servers with no token at a dispatcher the drift for $j \geq 1$ is given by

$$\begin{aligned}\frac{ds_{0,j}(t)}{dt} &= \vec{s}_{1,j}(t)t^* - \lambda q_0(t)s_{0,j}(t) + \lambda r(s_{0,j+1}(t) - s_{0,j}(t)) \\ &\quad + \vec{s}_{1,0}t^*(\hat{q}_{j-1}(t)^d - \hat{q}_j(t)^d).\end{aligned} \tag{19}$$

6.2. Drift equations with LCFS token selection

We now present a model for JIQ-SQ(d) without token withdrawals and LCFS token selection. The model generalizes the model in [9, Section IV.B] by allowing $d > 1$ and by considering phase-type distributed job sizes. We discuss the differences between the drift equations in Section 6.1 when the FCFS token selection strategy is replaced by LCFS. In this case the position j is such that new tokens are inserted in position 1, while all other tokens move back one position. The drift equations for $\hat{q}_i(t)$ are not affected by the token selection strategy. The drift for $\vec{s}_{i,j}(t)$ also remains the same, except that an additional term must be added because the position of a token can now increase by one when some other server places a token at the same server. This implies that the following term must be added to the drift of $\vec{s}_{i,j}(t)$ in (15):

$$r\vec{s}_{1,0}(t)t^*(1[j>1]\hat{q}_{j-1}(t)^{d-1}\vec{s}_{i,j-1}(t) - \hat{q}_j(t)^{d-1}\vec{s}_{i,j}(t)). \tag{20}$$

The drift for $\vec{s}_{1,j}(t)$ in (16) requires the same change as we need to add

$$r\vec{s}_{1,0}(t)t^*(1[j>1]\hat{q}_{j-1}(t)^{d-1}\vec{s}_{1,j-1}(t) - \hat{q}_j(t)^{d-1}\vec{s}_{1,j}(t)),$$

due to possible new tokens that are placed by servers becoming idle. Equation (17) and (18) do not change as it concerns servers without a token. Finally the drift of $s_{0,j}(t)$ given in (19) changes in the following ways. First, when $j > 1$ the last term must be removed as all new tokens are placed in position 1. This implies that for $j = 1$ we must add the term

$$\vec{s}_{1,0}(t)t^*$$

instead. Second a token can also be pushed backward by a new token which adds the term

$$r\vec{s}_{1,0}(t)t^*(1[j>1]\hat{q}_{j-1}(t)^{d-1}s_{0,j-1}(t) - \hat{q}_j(t)^{d-1}s_{0,j}(t)).$$

6.3. The fixed point with FCFS token selection

In this section we explain how to quickly compute the fixed point of the drift equations given by (14)–(19) when FCFS token selection is used. Summing (14) for i from $j \geq 1$ to ∞ implies that for any fixed point we have

$$\hat{q}_j = \frac{\vec{s}_1 t^*}{\lambda}\hat{q}_{j-1}^d,$$

which implies that $\hat{q}_j = (1-q_0)^{\sum_{i=0}^{j-1} d^i}$ with $q_0 = 1 - \vec{s}_1 t^*/\lambda$. This expression simplifies to $\hat{q}_j = (1-q_0)^j$ for $d = 1$ and $\hat{q}_j = (1-q_0)^{(d^j-1)/(d-1)}$ for $d > 1$.

Assume for now that we know q_0. If we focus on the fixed point equations for the variables $\vec{s}_{i,j}(t)$ for $i \geq 1$, $s_{0,j}(t)$ and $\vec{s}_{i,0}(t)$ with q_0 and $\hat{q}_j$ fixed, we get a system of linear equations. This allows us to construct a Quasi-Birth-Death (QBD) Markov chain [18] with state space $\{(i,j)|i \geq 1, j \geq 0\} \cup \{(0,j)|j \geq 1\}$ such that its stationary vector corresponds to the solution of this linear system of equations. As before state (i, j) corresponds to a server with i jobs, while its token is located in position j at some dispatcher for $j > 0$. State $(i, 0)$ correspond to a server with i jobs and no outstanding token. The rate matrix of this chain has the following form:

$$Q^{(FCFS)} = \begin{bmatrix} B_0 & B_1 & & & \\ C & A_0 & A_1 & & \\ & A_{-1} & A_0 & A_1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix}, \tag{21}$$

where the first block row corresponds to the states with $i = 0$, the second to the states with $i = 1$ and so on. To determine the steady state numerically, we truncate the value of j at some value B such that the matrix B_0 has size B and A_0 has size $B + 1$. An appropriate value of B can be selected in advance as it suffices to pick B such that $\hat{q}_B < \epsilon$ for ϵ sufficiently small. In the numerical experiments we set $\epsilon = 10^{-20}$.

The number of jobs in a server can increase in two ways. Either because a job is assigned by an empty dispatcher (at rate λq_0) or because the token of the server is in position one and a job arrives in that server (at rate λr). As a result we have

$$A_1 = (\lambda q_0 I_{B+1} + \lambda r e_2 e_1^*) \otimes I_m,$$

where e_j and e_k^* are the j-th column and k-th row of the unity matrix of the appropriate dimension, respectively. The $e_2 e_1^*$ indicates that the token is initially in position one and is removed after it is used. The I_m matrix is a size m unity matrix, where m is the number of service phases, and indicates that the service phase does not change. The matrix A_{-1} corresponds to a job completion, in which case a new job enters service and the token remains in the same position. Hence,

$$A_{-1} = I_{B+1} \otimes t^* \alpha,$$

with $t^* = (-T)e$. The state (i, j) with $j > 1$ also changes when a token moves a position forward (at rate λr), which decreases j by one. It can also change due to a change in the server phase. We therefore obtain

$$A_0 = \lambda r \begin{bmatrix} 0_{2m,m} & 0_{2m,(B-1)m} & 0_{2m,m} \\ 0_{(B-1)m,m} & I_{(B-1)m} & 0_{(B-1)m,m} \end{bmatrix} + I_{B+1} \otimes T$$
$$+ \begin{bmatrix} -\lambda q_0 I_m & 0_{m,Bm} \\ 0_{Bm,m} & -\lambda(r + q_0) I_{Bm} \end{bmatrix},$$

with $0_{k,\ell}$ a size $k \times \ell$ matrix with all entries equal to zero. The $2m$ zero rows in the first matrix are due to the requirement that $j > 1$ and the third matrix simply guarantees that the row sums are equal to zero. The matrix B_1 captures idle servers that become busy. This can happen in two ways. A job can be assigned because the token was in position one. In this case the token is removed (meaning j becomes 0) or a job can be assigned by an dispatcher without tokens (at rate λq_0):

$$B_1 = \begin{bmatrix} \lambda r & \\ 0_{B-1,1} & \lambda q_0 I_B \end{bmatrix},$$

where the matrix I_B indicates that the position does not change in the latter event. The state of an idle server can also change when its token moves forward:

$$B_0 = -\lambda(r + q_0)I_B + \lambda r \begin{bmatrix} 0_{1,B-1} & 0 \\ I_{B-1} & 0_{B-1,1} \end{bmatrix}.$$

Finally, the matrix C captures busy servers that become idle. When these servers have a token, it simply remains in place, otherwise the server places a token. Such a token is placed in position j with probability $\hat{q}_{j-1}^d - \hat{q}_j^d$, hence

$$C = \begin{bmatrix} 1 - \hat{q}_1^d & \hat{q}_1^d - \hat{q}_2^d & \dots & \hat{q}_{B-1}^d - \hat{q}_B^d \\ & I_B & & \end{bmatrix} \otimes t^*,$$

where entry j on the first row equals $\hat{q}_{j-1}^d - \hat{q}_j^d$. Notice that this first row is the only place in which the truncation of j has an effect and for $\hat{q}_B^d$ sufficiently small, the first row of C is numerically a stochastic row vector.

The stationary vector π of a QBD Markov chain has a matrix geometric form [18], which implies that for $k \geq 1$

$$\pi_k = \pi_1 R^{k-1},$$

with R the smallest non-negative solution to $A_1 + RA_0 + R^2 A_{-1} = 0$ and π_k contains the steady state probabilities belonging to the states with $i = k$ jobs in the server. For the two boundary vectors π_0 and π_1 we have

$$[\pi_0 \quad \pi_1] \begin{bmatrix} B_0 & B_1 \\ C & A_0 + RA_{-1} \end{bmatrix} = 0,$$

normalized such that $[\pi_0 \quad \pi_1(I - R)^{-1}]e = 1$.

The problem that remains is finding the value of q_0. Looking at the above Markov chain it is clear that increasing q_0 increases the rate to add a job. As a result the sum of the probabilities in the vector π_0 decreases as q_0 increases. Now for any fixed point of our set of ODEs we must have that $\pi_0 e = 1 - \lambda$ (as this must be the probability that a random server is idle). Hence, we can simply perform a bisection algorithm on $[0, 1]$ to find the unique q_0 where we compute $\pi_0 e$ of a QBD Markov chain during each iteration. Once the correct q_0 is found, the mean response time can be expressed as

$$\pi_1 \sum_{k \geq 1} k R^{k-1} e/\lambda = \pi_1 (I - R)^{-2} e/\lambda,$$

due to the matrix geometric form and Little's law.

This numerical approach is very effective for $d > 1$ as the probabilities $\hat{q}_j$ decrease doubly exponential in such case. This implies that we can use a small

B value and compute a fixed point of the set of ODEs in a fraction of a second. When $d = 1$ the values of $\hat{q}_j$ only decrease exponentially, meaning larger values of B are required. This is mostly the case when $(1 - \lambda)r$ is large as the mean number of tokens in a dispatcher exceeds this value for JIQ-SQ(d) without token withdrawals. Hence, for $d = 1$ and high loads this approach is still fast (i.e., for $r = 10$ and $\lambda > 0.8$), but a more efficient approach is still desirable when $d = 1$ and λ is smaller.

We therefore propose a second numerical approach that flips the order of the variables (i, j), such that state (j, i) corresponds to a server with i jobs and a token in position j, for $j > 0$, and no token when $j = 0$. The rate matrix of this Markov chain has the following form:

$$\bar{Q}^{(FCFS)} = \begin{bmatrix} \bar{B}_0 & \bar{B}_1 & \bar{B}_2 & \bar{B}_3 & \dots \\ \bar{C} & \bar{A}_0 & & & \\ & \bar{A}_{-1} & \bar{A}_0 & & \\ & & & \ddots & \ddots \end{bmatrix}, \tag{22}$$

as the position of a token cannot increase once it is placed. In this case we need to truncate the range of i to some value B. A sensible choice is to first compute the queue length distribution of a server for JIQ-SQ(d) *with* withdrawals and select B such that the probability of exceeding B can be neglected (e.g., is below 10^{-15}). Note that there are Bm rows that correspond to the first block row as any server without a token cannot be idle, while there are $Bm + 1$ rows for all other block rows. The expression for the size $Bm \times Bm + 1$ matrices $\bar{B}_j$, for $j > 0$, is given by

$$\bar{B}_j = \begin{bmatrix} t^* & 0_{m,Bm} \\ 0_{(B-1)m,1} & 0_{(B-1)m,Bm} \end{bmatrix} (\hat{q}_{j-1}^d - \hat{q}_j^d),$$

as they correspond to placing a token. The matrix $\bar{A}_{-1}$ corresponds to the token moving a position forward, meaning $\bar{A}_{-1} = \lambda r I_{Bm+1}$. The matrices $\bar{A}_0$ and $\bar{B}_0$ and $\bar{C}$ are slightly more involved, but not hard to obtain.

Let $\bar{\pi}$ be the invariant distribution of $\bar{Q}^{(FCFS)}$. Due to the structure of the rate matrix we have

$$\bar{\pi}_0 \left(\bar{B}_0 + \sum_{j \geq 1} \bar{B}_j ((-\bar{A}_0)^{-1} \bar{A}_{-1})^{j-1} (-\bar{A}_0)^{-1} \bar{C} \right) = 0, \tag{23}$$

$$\bar{\pi}_k = \bar{\pi}_0 \left(\sum_{j \geq k} \bar{B}_j ((-\bar{A}_0)^{-1} \bar{A}_{-1})^{j-k} (-\bar{A}_0)^{-1} \right), \tag{24}$$

as the matrix between brackets in (23) is the rate matrix of the Markov chain censored to the first Bm states and the matrix between brackets in (24) holds the mean time spend in the states with $j = k$ in between two visits to a state with $j = 0$. This means that

$$\sum_{k\geq 1} \bar{\pi}_k = \bar{\pi}_0 \left(\sum_{j\geq 1} \bar{B}_j \sum_{k=0}^{j-1} ((-\bar{A}_0)^{-1}\bar{A}_{-1})^k (-\bar{A}_0)^{-1} \right), \tag{25}$$

normalized such that $\bar{\pi}_0 e + \sum_{k\geq 1} \bar{\pi}_k e = 1$. The probability that a server is idle is given by the first entry of $\sum_{k\geq 1} \bar{\pi}_k$ and must equal $1 - \lambda$ when q_0 is properly set. This can be used to perform a simple bisection algorithm on $[0, 1]$ to find q_0.

6.4. The fixed point with LCFS token selection

Here we explain how the approach used to compute the fixed point for FCFS token selection can be modified such that it can be used to compute a fixed point for LCFS token selection. As explained in Section 6.2 the drift equations for $\hat{q}_j$ remain the same as in the FCFS setting. This allows us to use a QBD Markov chain with a rate matrix that is identical to $Q^{(FCFS)}$, except that the matrices A_0, B_0 and C are replaced by some matrices $\tilde{A}_0, \tilde{B}_0$ and $\tilde{C}$, respectively.

The matrix C is replaced by

$$\tilde{C} = \begin{bmatrix} e_1 \\ I_B \end{bmatrix} \otimes t^*,$$

as a token is always placed in position one when LCFS token selection is used. The matrices $\tilde{A}_0$ and $\tilde{B}_0$ can be written as A_0 and B_0 with an additional term, respectively, because the position of a token can now also increase by one when a new token is placed at the dispatcher holding the token. The rate at which servers empty is given by $\vec{s}_1 t^*$. As $q_0 = 1 - \vec{s}_1 t^*/\lambda$ holds for any fixed point (see Section 6.1), the rate $\vec{s}_1 t^*$ can be written in terms of q_0 as $\lambda(1 - q_0)$. This implies that the rate at which a token moves from position j to $j + 1$ can be expressed as $\lambda(1 - q_0) r \hat{q}_j^{d-1}$ (see (20)). Hence,

$$\tilde{B}_0 = B_0 + \lambda(1 - q_0) r \begin{bmatrix} -\hat{q}_1^{d-1} & \hat{q}_1^{d-1} & & \\ & -\hat{q}_2^{d-1} & \hat{q}_2^{d-1} & \\ & & \ddots & \ddots \end{bmatrix} \otimes I_m,$$

while

$$\tilde{A}_0 = A_0 + \lambda(1 - q_0) r \begin{bmatrix} 0 & 0 & & & \\ & -\hat{q}_1^{d-1} & \hat{q}_1^{d-1} & & \\ & & -\hat{q}_2^{d-1} & \hat{q}_2^{d-1} & \\ & & & \ddots & \ddots \end{bmatrix} \otimes I_m.$$

Apart from the changes to the matrices A_0, B_0 and C, the computation of the fixed point proceeds in exactly the same manner as in the FCFS case. As in the FCFS case this procedure is very fast for $d > 1$ and still efficient from $d = 1$

and λ close to one (say $\lambda > 0.8$). For $d = 1$ and smaller λ we can also flip the order of the states (i, j) as in the FCFS. However, contrary to the FCFS case, the position of a token always starts in one and can increase by one. When $d = 1$ the flipped Markov chain for the LCFS token selection is also a QBD Markov chain with rate matrix

$$\bar{Q}^{(LCFS)} = \begin{bmatrix} \bar{B}_0 & \bar{B}_1^* & & & \\ \bar{C} & \bar{A}_0^* & \bar{A}_1^* & & \\ & \bar{A}_{-1} & \bar{A}_0^* & \bar{A}_1^* & \\ & & \ddots & \ddots & \ddots \end{bmatrix}.$$

The matrices $\bar{A}_{-1}, \bar{B}_0$ and $\bar{C}$ are the same as in the FCFS case. The matrix $\bar{B}_1^*$ is given by

$$\bar{B}_1^* = \begin{bmatrix} t^* & 0_{m,Bm} \\ 0_{(B-1)m,1} & 0_{(B-1)m,Bm}, \end{bmatrix}$$

as a token is placed in position one when a busy server becomes idle. $\bar{A}_1^* = \lambda(1 - q_0)rI_{Bm+1}$ as $\lambda(1 - q_0)r$ equals the rate at which a token is pushed back by one position when $d = 1$. Finally, the matrix $\bar{A}_0^* = \bar{A}_0 - \bar{A}_1^*$.

7. Validation

In this section we present some simulation results to validate the mean field models presented in Section 6 for JIQ-SQ(d) without token withdrawals and FCFS or LFCS token selection. The parameter setting were chosen arbitrarily and are presented in Table 3. As in Table 2 we considered both exponential and hyper-exponential job sizes. The parameters of the order 2 hyper-exponential distribution are again fully determined by matching the mean (to one), the SCV and $f = p_1/\mu_1$. The results in Table 3 illustrates that the error of the mean field model seems to decrease to zero as the number of servers n tends to infinity.

8. Performance comparison

Having developed mean field models for JIQ-SQ(d) with and without withdrawals, we are now in a position where we can compare the performance of both variants. We should first of all note that the variant with withdrawals requires more work and this work is on the so-called critical path, as an idle server just received a job from a different dispatcher than the one that is holding its token. Hence, the main question is whether the additional effort to withdraw tokens helps the performance.

In Figures 5 and 6 we present the ratio between the mean response time of JIQ-SQ(d) with withdrawals and the mean response time of JIQ-SQ(d) without withdrawals and FCFS token selection as a function of λ for various choices of r and d. Figure 5 focuses on exponential job sizes, while Figure 6 considers more variable hyper-exponential job sizes. Various conclusions can be drawn from these figures:

settings	Job sizes	n	q_0	$E[R]$	rel.err. %
$\lambda = 0.8$, $r = 10$, $d = 2$, LCFS	Exponential	50	0.1310	1.1703	0.0319
		500	0.1269	1.1366	0.0021
		5000	0.1275	1.1339	-0.0002
		∞	0.1277	1.1341	
	HypExp $SCV = 10$ $f = 1/2$	50	0.1330	2.0014	0.2090
		500	0.1291	1.6838	0.0171
		5000	0.1287	1.6558	0.0002
		∞	0.1290	1.6555	
$\lambda = 0.65$, $r = 5$, $d = 1$, FCFS	Exponential	50	0.3070	1.3433	0.0244
		500	0.3366	1.3743	0.0018
		5000	0.3394	1.3765	0.0002
		∞	0.3398	1.3768	
	HypExp $SCV = 10$ $f = 1/10$	50	0.3140	2.6660	0.0138
		500	0.3418	2.7025	0.0003
		5000	0.3450	2.7020	0.0005
		∞	0.3452	2.7033	

Table 3: JIQ-SQ(d) without replacement: approximation error of the simulated q_0 and $E[R]$ and relative error of $E[R]$.

1. When the load is not too large, withdrawing tokens helps. The gain is more pronounced when r is small, $d = 1$ and for larger SCV. This is due to the fact that when the load is not too large, the likeliness of selecting a dispatcher without tokens is small and each token is guaranteed to correspond to an idle server when we withdraw tokens. We should however note that when $d > 1$ and r is not too small, the gain is very limited.
2. When the load is high, it is better not to withdraw tokens. This makes sense as the likeliness of selecting a dispatcher without tokens is high when the load is high. Therefore it is better to assign the job to a server that was known to be idle some time ago (when it placed its token), instead of assigning the job at random (as a random server under high load may contain many jobs).
3. When looking at the limit when λ approaches one, we see that the ratio of the response time between JIQ-SQ(d) with and without token withdrawals becomes independent of d. This can be understood by noting that when the load approaches one, there are very few tokens at the dispatchers, so even setting $d = 1$ implies that a dispatcher without tokens is selected with high probability when a server becomes idle.

With the third remark in mind, one may wonder how much gain we can still expect as λ tends to one compared to simple random assignment, as dispatchers with tokens become scarce when the load is very high so most jobs are assigned at random. Under random assignment, all servers behave as M/G/1-queues with load λ (as the mean service time is set to one). The mean field model for JIQ-SQ(d) with withdrawals suggested that the servers also behave as an

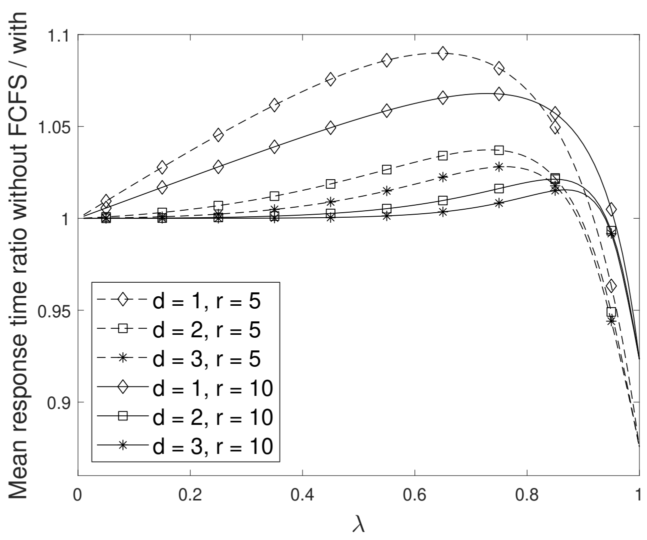

Figure 5: Ratio between mean response time of JIQ-SQ(d) without and with withdrawals for exponential job sizes as a function of the arrival rate λ for various choices of $r = n/m$ and d.

M/G/1-queue, but with a load given by λq_0. Now in the heavy traffic limit, the probability of having two tokens at the same dispatcher should tend to zero. As there are on average $r(1-\lambda)$ tokens residing at a single dispatcher, the probability q_0 should converge to $1 - r(1-\lambda)$ as λ tends to one. This suggests that the limit of the ratio between the mean response time of random assignment and JIQ-SQ(d) with withdrawals is given by

$$\lim_{\lambda \to 1^-} \frac{1 + \lambda E[G^2]/2(1-\lambda)}{1 + \lambda q_0 E[G^2]/2(1-\lambda q_0)} = \lim_{\lambda \to 1^-} \frac{(1-\lambda q_0)}{q_0(1-\lambda)}$$
$$= \lim_{\lambda \to 1^-} \frac{1 - \lambda + r(1-\lambda)\lambda}{(1-\lambda)} = r + 1.$$

as $q_0 \approx 1 - r(1-\lambda)$ for λ close to one. In other words, in heavy traffic JIQ-SQ(d) with token withdrawals is expected to perform $r+1$ times as good as random assignment even though nearly all jobs are assigned at random. This is confirmed by Figure 7 which illustrates that the limiting ratio tends to $r+1$ irrespective of the value of d or the job size distribution. The limiting ratio between random assignment and JIQ-SQ(d) without token withdrawals is harder to characterize as the comparison in Figure 6 implies that this ratio will exceed $r+1$ and will depend on the shape of the job size distribution.

We end this section by investigating the impact of the token selection method for JIQ without token withdrawals. Recall that with token withdrawals, this method has no impact as all tokens correspond to idle servers. In [9] some numerical results were presented that suggest that LCFS token selection achieves a slightly lower mean response time than FCFS. The paper also stated that

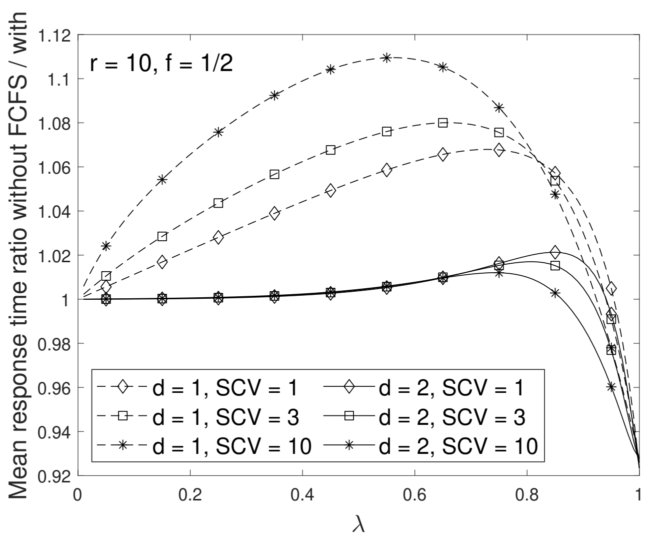

Figure 6: Ratio between mean response time of JIQ-SQ(d) without and with withdrawals for hyper-exponential job sizes with balanced means as a function of the arrival rate λ for $r = 10$ and various choices of SCV and d.

this can be expected as a token selected using LCFS is a more recent token and therefore is more likely to correspond to an idle server. However, one might also say that under LCFS some tokens may spend a long time at the dispatcher, which reduces their odds to correspond to an idle server. So the superior performance of LCFS token selection is not completely obvious.

In Figure 8 we plot the ratio of the mean response time of JIQ-SQ(d) without withdrawals and LCFS token selection, divided by the mean response time when the LCFS selection is replaced by FCFS selection and this for $d = 1, 2$, SCV=$1, 3, 10$ and λ ranging from 0 to 1. We note that LCFS indeed outperforms FCFS token selection in all cases, but the gain is quite limited: less than 2.5% for $d = 1$ and below 1% for $d = 2$.

9. Conclusions

In this paper we indicated that the well-known model introduced in [1] is not asymptotically exact for JIQ-SQ(d) with or without token withdrawals. We introduced a number of performance models for JIQ-SQ(d) load balancing with and without token withdrawals. Simulation experiments suggest that these models provide asymptotically exact results.

For JIQ-SQ(d) with token withdrawals these models suggest that the response time distribution of a job becomes identical to that in an M/PH/1 queue with load λq_0 in the large-scale limit. The value of q_0 depends on λ, d and r, but is independent of the job size distribution (with mean 1). The token selection method used by the dispatcher does not impact the system performance.

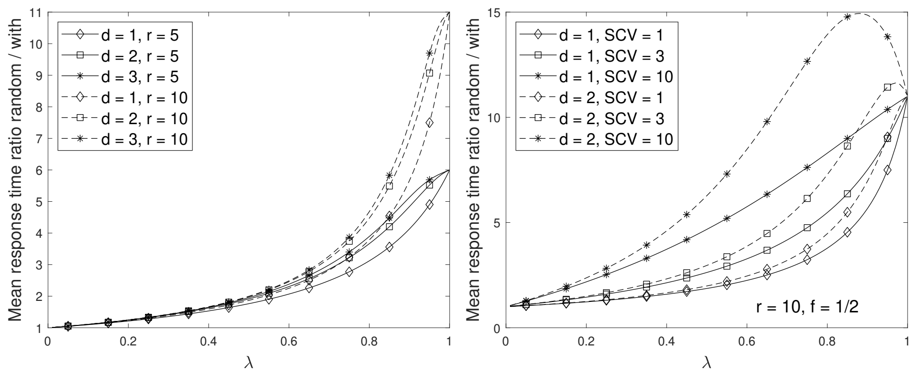

Figure 7: Ratio between mean response time of random assignment and JIQ-SQ(d) with withdrawals for exponential job sizes as a function of the arrival rate λ for various choices of $r = n/m$ and d (left) and forfor hyper-exponential job sizes with balanced means for $r = 10$ and various choices of SCV and d (right).

For JIQ-SQ(d) without token withdrawals the mean field model server dynamics are more complicated and depend on the token selection method used at the dispatcher. However the difference in performance between FCFS and LCFS token selection is quite limited and best for LCFS. When comparing JIQ-SQ(d) with and without token withdrawals, it is fair to state that the additional effort of withdrawing tokens mostly seems worthwhile for low to mean loads when $d = 1$ and the job sizes are highly variable.

References

[1] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services," *Perform. Eval.*, vol. 68, pp. 1056–1071, 2011.

[2] B. Van Houdt, "On the performance evaluation of distributed join-idle-queue load balancing," in *Proceedings of MASCOTS 2023*, 2023.

[3] A. Stolyar, "Pull-based load distribution in large-scale heterogeneous service systems," *Queueing Systems*, vol. 80, no. 4, pp. 341–361, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11134-015-9448-8

[4] ——, "Pull-based load distribution among heterogeneous parallel servers: the case of multiple routers," *Queueing Systems*, vol. 85, pp. 31–65, 2017.

[5] S. Foss and A. L. Stolyar, "Large-scale join-idle-queue system with general service times," *Journal of Applied Probability*, vol. 54, no. 4, p. 995–1007, 2017.

[6] C. Wang, C. Feng, and J. Cheng, "Distributed join-the-idle-queue for low latency cloud services," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2309–2319, 2018.

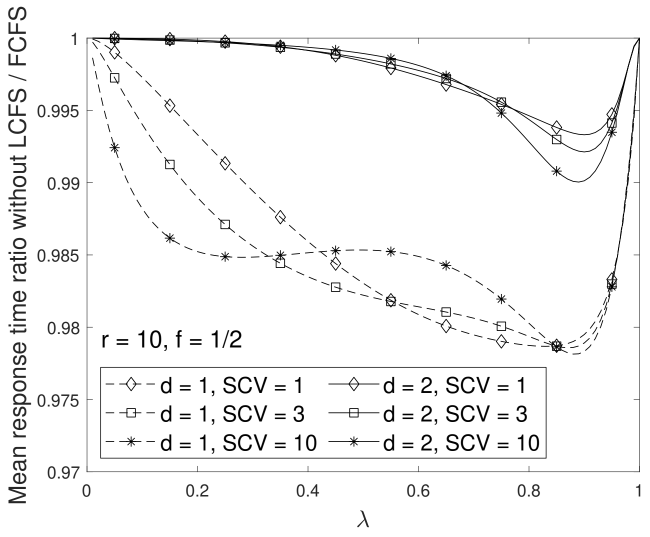

Figure 8: Ratio between mean response time of JIQ-SQ(d) without withdrawals and LCFS token selection versus FCFS token selection for hyper-exponential job sizes with balanced means as a function of the arrival rate λ for $r = 10$ and various choices of SCV and d.

[7] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, University of California, Berkeley, 1996.

[8] N. Vvedenskaya, R. Dobrushin, and F. Karpelevich, "Queueing system with selection of the shortest of two queues: an asymptotic approach," *Problemy Peredachi Informatsii*, vol. 32, pp. 15–27, 1996.

[9] M. Mitzenmacher, "Analyzing distributed join-idle-queue: A fluid limit approach," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 312–318.

[10] T. Kurtz, *Approximation of population processes.* Society for Industrial and Applied Mathematics, 1981.

[11] X. Zhou, F. Wu, J. Tan, Y. Sun, and N. Shroff, "Designing low-complexity heavy-traffic delay-optimal load balancing schemes: Theory to algorithms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, dec 2017. [Online]. Available: https://doi.org/10.1145/3154498

[12] M. van der Boor, S. Borst, and J. van Leeuwaarden, "Hyper-scalable JSQ with sparse feedback," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 1, pp. 1–37, 2019.

[13] S. Vargaftik, I. Keslassy, and A. Orda, "LSQ: Load balancing in large-scale heterogeneous systems with multiple dispatchers," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, p. 1186–1198, Jun. 2020. [Online]. Available: https://doi.org/10.1109/TNET.2020.2980061

[14] X. Zhou, N. Shroff, and A. Wierman, "Asymptotically optimal load balancing in large-scale heterogeneous systems with multiple dispatchers," *Performance Evaluation*, vol. 145, p. 102146, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166531620300663

[15] M. van der Boor, S. Borst, and J. van Leeuwaarden, "Load balancing in large-scale systems with multiple dispatchers," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[16] D. Mukherjee, S. Dhara, S. C. Borst, and J. van Leeuwaarden, "Optimal service elasticity in large-scale distributed systems," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, jun 2017. [Online]. Available: https://doi.org/10.1145/3084463

[17] D. Mukherjee and A. Stolyar, "Join idle queue with service elasticity: Large-scale asymptotics of a nonmonotone system," *Stochastic Systems*, vol. 9, no. 4, pp. 338–358, 2019.

[18] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods and stochastic modeling.* Philadelphia: SIAM, 1999.

[19] J. Kriege and P. Buchholz, *PH and MAP Fitting with Aggregated Traffic Traces.* Cham: Springer International Publishing, 2014, pp. 1–15. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05359-2_1

[20] F. Gillent and G. Latouche, "Semi-explicit solutions for M/PH/1-like queuing systems," *European Journal of Operational Research*, vol. 13, no. 2, pp. 151–160, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377221783900772