Replicate to the shortest queues

Rami Atar[*†] Isaac Keslassy [*] Gal Mendelson[*†]

Abstract

This paper introduces a load balancing policy that interpolates between two well known policies, namely *join the shortest queue* (JSQ) and join the least workload (JLW), and studies it in heavy traffic. This policy, which we call *replicate to the shortest queues* (RSQ(d)), routes jobs from a stream of arrivals into buffers attached to N servers by replicating each arrival into $1 \le d \le N$ tasks and sending the replicas to the d shortest queues. When the first of the tasks reaches a server, its $d-1$ replicas are canceled. Clearly RSQ(1) is equivalent to JSQ, and it has been shown that RSQ(N) is equivalent to JLW; intermediate values of d provide tradeoff between good performance measures of JSQ and those of JLW. In heavy traffic, a key property underlying asymptotic analysis of load balancing policies is *state space collapse* (SSC). Unlike policies such as JSQ, where SSC is well-understood, the treatment of SSC under RSQ(d) requires to address the massive cancellations that highly complicate the queue length dynamics. Our first main result is that SSC holds under RSQ(d) for possibly heterogeneous servers. Based on this result we obtain diffusion limits for the queue lengths in the form of one-dimensional reflected Brownian motion, asymptotic characterization of the short-time-averaged delay process, and a version of Reiman's snapshot principle. We illustrate using simulations that, as d increases, the server workloads become more balanced, and the delay distribution's tail becomes lighter. We also discuss the implementation complexity of the policy as compared to that of the *redundancy routing* policy, to which it is closely related.

AMS subject classification: 60F17, 60J60, 60K25.

Keywords: Randomized load balancing, replicate to shortest queues, join the shortest queue, join the least workload, task redundancy, job cancellations, diffusion limits, heavy traffic, state space collapse

1 Introduction

We consider a queueing model in which a single dispatcher routes incoming jobs to N heterogeneous servers, each with its own buffer in which a queue can form. Three well known load balancing algorithms for this model are *join the shortest queue* (JSQ), *redundancy routing* (R(d)) and *join the least workload* (JLW). In JSQ, jobs are sent to the buffer with the shortest queue. In R(d), jobs are replicated into $1 < d \le N$ *tasks*, which are routed into d randomly chosen buffers, and when the first of the tasks starts service, its replicas are canceled and removed from the system (this is referred to as *early cancellation*, as opposed to *late cancellation* where cancellation occurs when a task first completes service). In JLW, jobs are sent to the buffer with the least work. This paper introduces and studies an algorithm that combines elements of JSQ and R(d), that we

[*]The Viterbi Faculty of Electrical Engineering, Technion–Israel Institute of Technology, Haifa 32000, Israel

[†]Research supported in part by the ISF (grant 1184/16)

call *replicate to the shortest queues* (RSQ(d)), where arriving jobs are replicated into d tasks, and these tasks are routed to the d shortest queues. The cancellation mechanism is as in R(d). This policy benefits from the good performance offered by both R(d) and JSQ while achieving better workload balance and lower job completion times (which we refer to as *delay*). Moreover, as we argue, RSQ(d) constitutes a range of policies from JSQ (when $d = 1$) to JLW (when $d = N$). When there is significance to the tradeoff between performance and implementation complexity, RSQ(d) provides a degree of freedom. Depending on the application, d can be chosen not too small so that delay performance is improved, and not too large so that replication overhead is tolerable. As we later argue, the implementation complexity of RSQ(d) is not worse than that of R(d). The main results of this paper are concerned with the analysis of this policy in heavy traffic, establishing state space collapse and diffusion limits. In addition, simulation results are provided and discussed.

Our motivation for studying RSQ(d) stems from the aforementioned tradeoff related to JSQ and JLW, but also from several advantages it has over JSQ and R(d). We provide simulation results showing that this policy improves several performance measures over these policies.

We start with some background on the aforementioned policies and then discuss how RSQ(d) stands in comparison to them. R(d) has been an active research topic due to its importance in applications. A large body of research is concerned with characterizing the impact of replicating jobs on various network performance criteria. Aside from the inherent robustness it offers with respect to network failures, the use of replication and cancellation in sophisticated schedulers is shown to reduce the average delay, as well as the tail of the delay distribution (*e.g.,* [1], [2], [17]). Analytic studies provide conditions under which R(d) is beneficial and characterize various system performance measures such as mean delay and long time behavior (*e.g.,* [12], [14], [20]). R(d) is also known to have good balancing properties with respect to workload. See [3] for a comprehensive review. JSQ is a well studied algorithm that is known to achieve the capacity region [9, 11] and provides several good performance guarantees, *e.g.,* [3, 7]. It is also widely used in practice [13]. The JLW policy achieves the capacity region [11] and provides excellent performance guarantees [8, 10, 23].

A significant disadvantage of R(d) (for $d < N$) is that it does not achieve the capacity region. That is, when servers are heterogeneous, this policy may be unstable even when the system is subcritical [11]. In particular, diffusion limits need not exist [3]. RSQ(1) and RSQ(N) clearly achieve the capacity region and we show in this paper that the diffusion limits exist for any d. While the property that RSQ(d) achieves the capacity region for $1 < d < N$ is not proved in this paper, we expect that it does, but leave this as an open problem.

While JSQ provides queue length balance, when servers are heterogeneous, the server workloads are not balanced. This has a negative effect on job completion times, as workload balance reduces the chance of a large waiting time [12]. Our simulation results indicate that using d as small as 2 leads to a significant improvement in terms of workload balance and job completion times.

As for JLW, in practice, the dispatcher may have no access to workload information. Still, it is possible to implement JLW by applying R(N). The equivalence of these two policies is heuristically obvious, and has recently been proved rigorously in [3]. Since RSQ(N) and R(N) are identical policies, the last comment applies to RSQ(N) as well. However, implementing JLW via R(N) may incur an intolerable overhead when N is large.

Our goal in this paper is to analyze RSQ(d) in heavy traffic. We obtain diffusion limit results for the queue length and also extract information on the delay. The model which we study accommodates server heterogeneity and allows for general renewal arrivals and service time distributions with finite second moment. The main step towards proving our results is establishing that *state space collapse* (SSC) holds for RSQ(d). Denoting by n the heavy traffic parameter, SSC refers to a balance achieved by the queue lengths at the different buffers that is of order $o(n^{1/2})$. It is known since Reiman's paper [19] that SSC holds for JSQ, and it has been shown to hold under other policies, and used to establish diffusion limit results (*e.g.,* [3], [6], [7]).

As opposed to policies with no replication and cancellation (*e.g.,* JSQ and JLW) where SSC is well understood, proving SSC for RSQ(d) presents as a mathematical challenge. In this present setting, queue lengths correspond to the number of tasks in the buffers, including tasks that are to be canceled. Because of these cancellations, the dynamics of the queueing processes are highly complicated. The queue lengths themselves do not constitute a state descriptor even in the special case where the arrivals follow a Poisson process and the service times are exponential. This is due to the fact that the state of the system must record the relations among the replicas residing in the buffers. An analysis of the full state process in heavy traffic appears to be notoriously hard. We show that despite these difficulties one may analyze the queue length and delay processes. Our first main result establishes SSC for the queue lengths.

Based on the SSC, we establish several additional results for this model. First we show that in the heavy traffic limit the queue lengths are given by a one-dimensional reflected Brownian motion (RBM). Then, toward analyzing the delay, we develop a version of Reiman's snapshot principle (RSP). The standard version of this principle asserts that the delay process, normalized at diffusion scale, converges to a scalar multiple of the normalized queue length limit [18]. Due to the cancellations involved in RSQ, it does not seem reasonable that RSP is valid in this form. However, we were able to prove that a *short time average* version of the delay process does converge to a process given by a scalar multiple of the normalized queue length limit. This weak version of the RSP provides, in particular, a characterization of the short time average delay process at the diffusion limit.

As far as tools for establishing SSC are concerned, we mention well known work by Bramson [6] and Williams [22] on which numerous papers on SSC for various queueing models have been based (*e.g.,* [7]). RSQ(d) cannot be treated via these tools because they do not accommodate task cancellation. The closest reference for the current paper is [3], where it was shown than a generalization of the SSC property, referred to as sub-diffusivity of the deviation process (SDDP), holds for several time varying queuing systems (not necessarily in heavy traffic) including JSQ, R(d) and JLW. In the special case of fixed arrival and service rates, SDDP reduces to SSC. Using a similar approach, we identify necessary conditions for the deviation of the different queue lengths from one another to exceed $o(n^{1/2})$, and argue that they occur with small probabilities. However, the technical details are model specific, and the development here is completely different from those of [3]. Specifically, R(d) was treated in [3] via its equivalence to JLW, without any analysis of the replication and cancellation mechanisms. Our approach is more direct.

The organization of this paper is as follows. This section concludes with the introduction of some notation that is used in the sequel. In §2 the model is described and the main results are stated. The proofs appear in §3. §4 contains a discussion of implementation complexity and simulation results.

Notation. For $a, b \in \mathbb{R}$, the maximum (resp., minimum) is denoted by $a \vee b$ (resp., $a \wedge b$). For $x \in \mathbb{R}^k$ (k a positive integer), $\|x\|$ denotes the ℓ_1 norm. Denote $\mathbb{R}_+ = [0, \infty)$, and for $f : \mathbb{R}_+ \to \mathbb{R}^k$,

$\|f\|_T = \sup_{t\in[0,T]} \|f(t)\|$, and, for $\theta > 0$,

$$w_T(f,\theta) = \sup_{0\le s<u\le s+\theta\le T} \|f_u - f_s\|.$$

For a Polish space $\mathcal{S}$, let $\mathbb{C}_{\mathcal{S}}(0,T)$ and $\mathbb{D}_{\mathcal{S}}(0,T)$ denote the set of continuous and, respectively, cadlag functions $[0,T] \to \mathcal{S}$. Write $\mathbb{C}_{\mathcal{S}}$ and $\mathbb{D}_{\mathcal{S}}$ for the case where $[0,T]$ is replaced by $\mathbb{R}_+$. Endow $\mathbb{D}_{\mathcal{S}}$ with the Skorohod J_1 topology. Write $X_n \Rightarrow X$ for convergence in distribution. A sequence of processes X_n with sample paths in $\mathbb{D}_{\mathcal{S}}$ is said to be *C-tight* if it is tight and every subsequential limit has, with probability 1, sample paths in $\mathbb{C}_{\mathcal{S}}$. For $m \in \mathbb{R}$ and $\sigma \in \mathbb{R}$, an (m,σ^2)-*Brownian motion* (BM) is a 1-dimensional BM starting from zero, having drift m and infinitesimal covariance σ^2. Given a time interval $J = [t_1,t_2] \subset \mathbb{R}_+$ we write $f[t_1,t_2] = f[J] = f(t_2) - f(t_1)$, for any function f defined on $\mathbb{R}_+$.

2 Model and results

A sequence of models indexed by $n \in \mathbb{N}$ is defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ as follows. A fixed number of servers, N, labeled by $\{1,\dots,N\}$, have unlimited buffers, one dedicated to each. Each server is non-idling and offers service on a first-come-first-served basis. There is a single stream of arriving jobs. The kth job to arrive (after time zero) is referred to as *job k*.

Upon arrival, each job is replicated into d $(1 < d \le N)$ identical replicas, called *tasks*, and sent to those d buffers that contain the least number of tasks at the moment. Ties are broken by prioritizing buffer i over buffer j if $i < j$. When a task is called to start service, all of its $d-1$ replicas are removed from the system. When the task is completed, it is removed from the system and this epoch is the time of completion of the corresponding job. Thus each job is processed by a single server and replication does not increase the total amount of processing work.

The stream of incoming jobs is modeled by a renewal process. Let parameters λ^n be given, representing the reciprocal mean inter-arrival time in the nth system. Let a sequence $\{IA(l), l \in \mathbb{N}\}$ of strictly positive *i.i.d.* RVs with mean 1 and variance $0 < V_{IA(1)} < \infty$ be given. Define

$$A(t) = \sup\{l : \sum_{k=1}^{l} IA(l) \le t\}.$$

The counting process for arrivals, A^n, is assumed to be given by $A^n(t) = A(\lambda^n t)$.

Let parameters μ_i^n, $i = 1,\dots,N$, be given, representing the reciprocal mean service times of server i in the nth system. For $i \in \{1,\dots,N\}$, let $\{T_i(l) : l \in \mathbb{N}\}$ be a sequence of strictly positive *i.i.d.* RVs with mean 1 and variance $0 < V_{T_i(1)} < \infty$. It is assumed that the kth task effectively served by server i takes $T_i^n(k) := T_i(k)/\mu_i^n$ units of time to process.

Denote by Q_i^n the queue length of tasks in buffer i (including the task in service). We emphasize that this includes tasks that will eventually be canceled. We assume the system starts empty. The $N+1$ processes $\{T_i(\cdot)\}$ and A are assumed to be mutually independent. These processes, the routing mechanism and the zero initial condition specified above uniquely define the queue length process $Q^n = (Q_1^n, \dots, Q_N^n)$.

As for the parameters introduced above, it is assumed that there exist constants $\lambda \in (0,\infty)$, $\hat{\lambda} \in \mathbb{R}$, $\mu_i \in (0,\infty)$ and $\hat{\mu}_i \in \mathbb{R}$, $i \in \{1,\dots,N\}$, such that

$$\lim_{n\to\infty} n^{-1/2}(\lambda^n - n\lambda) = \hat{\lambda}, \tag{1}$$

and

$$\lim_{n\to\infty} n^{-1/2}(\mu_i^n - n\mu_i) = \hat{\mu}_i, \qquad i = 1, \dots, N. \tag{2}$$

Moreover, a critical load condition is assumed. The first order terms in the n-scale arrival rate and total processing rate are given by λ and $\sum_{j=1}^N \mu_j$, respectively. Therefore, criticality corresponds to

$$\lambda = \sum_{i=1}^N \mu_i, \tag{3}$$

a condition assumed throughout.

Let $\hat{Q}_i^n = n^{-1/2} Q_i^n$ denote the diffusion scale version of the queue length processes. The main result of this paper is the following.

Theorem 2.1 (State Space Collapse)**.** *As $n \to \infty$,*

$$\max_{1\le i,j\le N} \|\hat{Q}_i^n - \hat{Q}_j^n\|_T \to 0 \textit{ in probability.}$$

We now use the SSC result to establish a weak limit result for the queue lengths. The Skorohod map Γ from $\mathbb{D}(\mathbb{R}_+ : \mathbb{R})$ to itself is defined by

$$\Gamma[\phi](t) = \phi_t - \inf_{s\le t} \phi_s \wedge 0, \qquad t \ge 0.$$

If $\{\beta_t\}$ is an (m, σ^2)-BM for some $m \in \mathbb{R}$ and $\sigma \in (0, \infty)$, then the process $\{\beta_t^0\}$ defined by the pathwise transformation $\beta^0 = \Gamma[\beta]$ is referred to as *an (m, σ^2)-reflecting Brownian motion (RBM).*

Denote $\hat{m}_0 = N^{-1}(\hat{\lambda} - \sum_i \hat{\mu}_i)$ and $\hat{\sigma}_0^2 = N^{-2}(\lambda V_{IA(1)} + \sum_{i=1}^N \mu_i V_{T_i(1)})$.

Corollary 2.2 (Diffusion limit)**.** *Let β^0 be a $(d\hat{m}_0, d^2\hat{\sigma}_0^2)$-RBM. Then, as $n \to \infty$,*

$$(\hat{Q}_1^n, \dots, \hat{Q}_N^n) \Rightarrow (\beta^0, \beta^0, \dots, \beta^0).$$

Note that the above results hold with no conditions on the μ_i's, except the heavy traffic condition (3).

We next discuss the delay experienced by jobs. Characterizing the behavior of delay under the current setting, by appealing to the queue length result, Corollary 2.2, is not at all straightforward. A link between the queue length process and the delay process at the diffusion scale, often used to compute the latter based on the former, is known as *Reiman's snapshot principle (RSP).* It asserts that the delay process, normalized at diffusion scale, converges to a scalar multiple of the normalized queue length limit (see [18], and *e.g.,* [4] for a recent application of this approach). However, the policy studied in this subsection involves massive cancellations, to the degree that it does not seem reasonable that a strong statement such as RSP is valid. Our goal is to argue that a *short time average* version of the delay process does converge to a process given by a scalar multiple of the normalized queue length limit, and in particular, to characterize this limit process.

Toward defining the average delay process, we need some notation. Recall that jobs are numbered by their order of arrival. Denote by

$$\alpha^n(k) = \inf\{t : A^n(t) \ge k\}, \qquad k \in \mathbb{N}$$

the arrival time of the kth job in the nth system. For integer $1 < d \le N$, let $\mathcal{B}_d := \{b \subset \{1,...,N\}, |b| = d\}$ be the set of all d-size subsets of $\{1,...,N\}$. For $k \in \mathbb{N}$, denote by $G^n(k)$ a RV taking values in $\mathcal{B}_d$, representing the collection of d servers to which job k's tasks were routed. Let $i^n(k)$ be a $\{1,\dots,N\}$-valued RV indicating the index of the server where the task was completed. Thus $i^n(k) \in G^n(k)$, and the members of $G^n(k) \setminus \{i^n(k)\}$ are those servers where the corresponding tasks were canceled. Denote by $\beta^n(k)$ the departure time of job k, that is, the departure time of job k's task performed at server $i^n(k)$. Let $J^n(k) = [\alpha^n(k), \beta^n(k)]$ denote the time interval at which at least one of job k's tasks are present in the system, and let $\delta^n(k) = \beta^n(k) - \alpha^n(k)$ denote job k's delay. Our result on delay refers to averaging over jobs that arrive within a short time interval, defined as follows. Let

$$K^n(t_1,t_2) = \{k \in \mathbb{N} : A^n(t_1) < k \le A^n(t_2)\}$$

denote the set of jobs to arrive during the interval $(t_1, t_2]$. Let

$$\mathit{\Delta}^n(t_1,t_2) = \frac{\sum_{k\in K^n(t_1,t_2)} \delta^n(k)}{|K^n(t_1,t_2)| \vee 1}, \qquad \hat{\mathit{\Delta}}^n(t_1,t_2) = n^{1/2}\mathit{\Delta}^n(t_1,t_2), \qquad 0 \le t_1 < t_2.$$

Theorem 2.3 (Delay). *Fix a sequence $a^n > 0$ with $a^n \to 0$, $n^{1/2}a^n \to \infty$ as $n \to \infty$, and denote*

$$\hat{\mathit{\Delta}}^n(t) = \hat{\mathit{\Delta}}^n(t, t+a^n), \qquad t \ge 0.$$

Let $\check{m}_1 = d^{-1}\lambda^{-1}N\hat{m}_0 = \lambda^{-1}(\hat{\lambda} - \sum_i \hat{\mu}_i)$, and $\check{\sigma}_1^2 = d^{-2}\lambda^{-2}N^2\hat{\sigma}_0^2 = \lambda^{-1}V_{IA(1)} + \lambda^{-2}\sum_i \mu_i V_{T_i(1)}$. Let β^1 be a $(\check{m}_1, \check{\sigma}_1^2)$-RBM. Then, as $n \to \infty$,

$$\hat{\mathit{\Delta}}^n \Rightarrow \beta^1.$$

In the above result the value of d does not affect the assumptions nor the limit. Heuristically, however, one expects that as d grows, the delay process should behave more regularly, because redundancy acts to balance delay. We leave open the question of how to quantify an improvement in the behavior of delay with an increase in d.

3 Proofs

In this subsection we analyze the RSQ model and prove Theorem 2.1, Corollary 2.2 and Theorem 2.3. The main difficulty that we face is that the cancellations that take place affect the queue length enormously but are hard to get control over. Cancellations at a given buffer are triggered by the state of all other buffers, and their dynamics does not follow a simple mechanism such as FIFO. Our technique does not rely on analyzing equations that fully characterize the cancellations, but only on equations that give partial information on them. These include certain balance equations, namely (7) and (12) below, and a rough bound on cancellation count in terms of departure count, namely (15) below. It is perhaps surprising that this limited information suffices for our purposes.

First we introduce several processes (such as potential service, departure and cancellation) necessary to write down the dynamics of the queue length process.

The potential service processes are constructed from the task sizes as

$$S_i^n(t) = \sup\Big\{l \ge 0 : \sum_{k=1}^{l} T_i^n(k) \le t\Big\}. \tag{4}$$

The value $S_i^n(t)$ gives the number of task departures from queue i by the time that the corresponding server has busy for t units of time. Note that S_i^n are renewal processes.

The cumulative idle time of server i at time t is denoted by $I_i^n(t)$. The non-idling property can be encoded by the condition

$$\int_{[0,\infty)} Q_i^n(t) dI_i^n(t) = 0, \qquad i = 1, \dots, N. \tag{5}$$

Let $\Theta_i^n(t) = t - I_i^n(t)$, $t \geq 0$ denote the cumulative busy time of server i at time t. By A_i^n, C_i^n and D_i^n we denote counting processes for task-arrivals into buffer i, number of task cancellations at buffer i and number of departures from buffer i, respectively. Here, and throughout, the term *departure* corresponds to the removal of a task due to service completion (as opposed to cancellation). The relation between the three processes D_i^n, S_i^n and Θ_i^n is expressed by

$$D_i^n(t) = S_i^n(\Theta_i^n(t)), \qquad t \geq 0, \, i = 1, \dots, N, \tag{6}$$

and the balance equation for the i-th queue is

$$Q_i^n(t) = A_i^n(t) - D_i^n(t) - C_i^n(t), \qquad i = 1, \dots, N. \tag{7}$$

Scaled (and centered) versions of the arrival and potential service processes are given by

$$\hat{A}^n(t) = n^{-1/2}(A^n(t) - \lambda^n t), \quad \hat{S}_i^n(t) = n^{-1/2}(S_i^n(t) - \mu_i^n t). \tag{8}$$

By the functional central limit theorem, the $N+1$ dimensional processes $(\hat{A}^n, \hat{S}_1^n, \dots, \hat{S}_N^n)$ converge to a BM with drift zero and a diagonal covariance matrix Σ with $\Sigma(1,1) = \lambda$ and $\Sigma(i+1, i+1) = \mu_i V_{T_i(1)}$, $i \geq 1$ (see Section 17 of [5]).

Fix $z \in (1, \infty)$ and denote

$$\tau^n = \inf\{t : \|\hat{Q}^n(t)\| \geq z - 1\}. \tag{9}$$

By definition, $\|\hat{Q}^n(t)\| \leq z - 1$ for all $t < \tau^n$. Taking into account the fact that the upward jumps of $\hat{Q}^n$ are bounded by $cn^{-1/2}$, where c is a constant, it follows that $\|\hat{Q}^n(t)\| \leq z$ for all $t \leq \tau^n$, where, throughout, one assumes without loss of generality that n is sufficiently large. Fix T. We prove the queue length processes undergo SSC in $[0, T]$ using a bootstrap argument. Given z, we prove SSC occurs up to time $\tau^n \wedge T$ using the bound on $\|\hat{Q}^n(t)\|$ and an upper bound of order $n^{-1/2}$ on the delay experienced by jobs (Lemma 3.1 below). We then use this SSC result to show $\lim_{z\to\infty} \limsup_n \mathbb{P}(\tau^n < T) = 0$, thus completing the proof. Let

$$\bar{\delta}^n = \sup\{\delta^n(k) : k \leq A^n(T)\}, \qquad \bar{\delta}_z^n = \sup\{\delta^n(k) : k \leq A^n(T \wedge \tau^n)\}, \tag{10}$$

denote the maximal delay among all jobs to arrive during the time intervals $[0, T]$ and $[0, T \wedge \tau^n]$, respectively. Denote $\mu_* = (\min_i \mu_i)/2$ and $\gamma_z = (z+1)\mu_*^{-1}$.

We first provide a rough bound on the delay experienced by jobs to arrive by time $T \wedge \tau^n$.

Lemma 3.1. *One has*

$$\lim_n \mathbb{P}(\bar{\delta}_z^n > \gamma_z n^{-1/2}) = 0.$$

Proof. For $k \in \mathbb{N}$, denote

$$\Omega^{n,k} = \{\alpha^n(k) \le T \wedge \tau^n, \delta^n(k) > \gamma_z n^{-1/2}\}.$$

Fix k and consider the event $\Omega^{n,k}$. For simplicity, write $\alpha^n = \alpha^n(k)$, $\beta^n = \beta^n(k)$, $i^n = i^n(k)$ and $J^n = J^n(k)$. Clearly, server i^n is busy during the time interval J^n. Hence, on the event $\Omega^{n,k}$,

$$\Theta^n_{i^n}(\beta^n) - \Theta^n_{i^n}(\alpha^n) = \beta^n - \alpha^n = \delta^n(k) > \gamma_z n^{-1/2}.$$

Moreover, the number of tasks that depart the queue i^n during the interval J^n cannot exceed the queue length at the time of job k's arrival, α^n. Thus on $\Omega^{n,k}$,

$$D^n_{i^n}[J^n] \le Q^n_{i^n}(\alpha^n) \le \|Q^n\|_{T\wedge\tau^n} \le zn^{1/2}.$$

Using (6), it follows that on the event $\Omega^{n,k}$ one has

$$S^n_{i^n}(\Theta^n_{i^n}(\alpha^n) + \gamma_z n^{-1/2}) - S^n_{i^n}(\Theta^n_{i^n}(\alpha^n)) \le zn^{1/2}.$$

Note that $\{\bar{\delta}^n_z > \gamma_z n^{-1/2}\} = \cup_{k\ge 1}\Omega^{n,k}$. Thus, on $\{\bar{\delta}^n_z > \gamma_z n^{-1/2}\}$, there exists $i \in \{1,\dots,N\}$ and $\sigma \in [0,T]$ such that

$$S^n_i(\sigma + \gamma_z n^{-1/2}) - S^n_i(\sigma) \le zn^{1/2}, \tag{11}$$

where we used the fact that $\alpha^n(k) \le T$ on the event $\Omega^{n,k}$. By the definition (8) of the scaled processes $\hat{S}^n_i$, (11) can be written as

$$\hat{S}^n_i(\sigma + \gamma_z n^{-1/2}) + (\sigma + \gamma_z n^{-1/2})\mu^n_i n^{-1/2} - \hat{S}^n_i(\sigma) - \sigma\mu^n_i n^{-1/2} \le z.$$

By (2), for n large enough $\mu^n_i \ge \mu_* n$. Hence the above implies

$$\gamma_z\mu_* - \max_i w_{2T}(\hat{S}^n_i, \gamma_z n^{-1/2}) \le z.$$

Since $\gamma_z\mu_* - z = 1$, we obtain

$$\mathbb{P}(\bar{\delta}^n_z > \gamma_z n^{-1/2}) \le \mathbb{P}(\max_i w_{2T}(\hat{S}^n_i, \gamma_z n^{-1/2}) \ge 1).$$

Since $\{S^n_i\}$ are C-tight, the RHS converges to zero as $n \to \infty$, proving the result. $\square$

Simple relations between cancellations and departures are as follows. At the time when a task is called for service, $d-1$ tasks are canceled elsewhere in the system. Therefore the number of cancellations by time t is equal to $d-1$ times the number of tasks admitted into service by that time. Note that the latter is equal to the number of departures plus the number of tasks in service at that time. Consequently,

$$\sum_{i=1}^N C^n_i(t) = (d-1)\sum_{i=1}^N D^n_i(t) + (d-1)M^n(t), \tag{12}$$

where

$$M^n(t) = \sum_i 1_{\{Q^n_i(t)>0\}} \tag{13}$$

is the number of non-idle servers at time t. Clearly,

$$0 \le M^n(t) \le N, \ t \ge 0. \tag{14}$$

During a time interval $J = [t_1, t_2]$, every cancellation in a server j corresponds to a task that has either departed another server i or is still being processed there at t_2. Hence for every j,

$$C_j^n[J] \le \sum_{i:i\ne j} D_i^n[J] + M^n(t_2). \tag{15}$$

Our main estimate toward proving Theorem 2.1, Corollary 2.2 and Theorem 2.3 is the following.

Lemma 3.2. *As $n \to \infty$,*

$$\max_{1\le i,j\le N} \|\hat{Q}_i^n - \hat{Q}_j^n\|_{T\wedge\tau^n} \to 0, \ \textit{in probability.}$$

Proof.

Fix $\eta > 0$. Consider the event that there exist two normalized queue lengths that drift 2η apart by the time $T \wedge \tau^n$, namely

$$\Omega^n = \{\max_{i,j} \|\hat{Q}_i^n - \hat{Q}_j^n\|_{T\wedge\tau^n} > 2\eta\}. \tag{16}$$

For any vector $v \in \mathbb{R}^N$ we use the notation $v_{(i)}$ for the ith smallest coordinate of v, counting multiplicity. That is, $\{v_{(i)}\}$ satisfy

$$v_{(1)} \le \cdots \le v_{(N)},$$

such that for each $i \in \{1, \ldots, N\}$, $v_{(i)} = v_j$ for exactly one j. Consider the first time when $\hat{Q}_{(N)}^n$ and $\hat{Q}_{(d)}^n$ are η apart,

$$t_a^n = \inf\{t : \hat{Q}_{(N)}^n(t) - \hat{Q}_{(d)}^n(t) > \eta\}, \tag{17}$$

and similarly, the first time when $\hat{Q}_{(d)}^n$ and $\hat{Q}_{(1)}^n$ are η apart,

$$t_b^n = \inf\{t : \hat{Q}_{(d)}^n(t) - \hat{Q}_{(1)}^n(t) > \eta\}. \tag{18}$$

Then, with

$$t_2^n = \min(t_a^n, t_b^n), \tag{19}$$

it is clear that $\Omega^n \subset \{t_2^n \le T \wedge \tau^n\}$. Hence

$$\mathbb{P}(\Omega^n) \le \mathbb{P}(\Omega_a^n) + \mathbb{P}(\Omega_b^n), \quad \Omega_a^n = \{t_b^n \ge t_a^n,\, t_2^n = t_a^n \le T\wedge\tau^n\}, \quad \Omega_b^n = \{t_b^n < t_a^n,\, t_b^n = t_2^n \le T\wedge\tau^n\}. \tag{20}$$

The argument proceeds by providing estimates on each of the terms on the RHS of (20). We slightly simplify the notation by writing τ, t_a, t_b, t_2 for τ^n, t_a^n, t_b^n, t_2^n, respectively.

Step 1: estimating $\mathbb{P}(\Omega_a^n)$. Consider the event Ω_a^n. On this event, $\hat{Q}_{(N)}^n$ and $\hat{Q}_{(d)}^n$ are η apart at time t_2, while $\hat{Q}_{(d)}^n$ and $\hat{Q}_{(1)}^n$ are still η close. Denote by i_{max} the index of the maximal queue length at t_2, that is, the (minimal) $i \in \{1, \ldots, N\}$ such that $\hat{Q}_i^n(t_2) = \hat{Q}_{(N)}^n(t_2)$. Let also

$$t_1 = \sup\{t < t_2 : A_{i_{max}}^n(t) < A_{i_{max}}^n(t_2)\}$$

be the last time before t_2 when a task arrived at queue i_{max}. Note that on the event Ω^n_a, $t_2 > 0$. By the definition of t_2, $\hat{Q}^n_{i_{max}}(t_2) > \eta$. Since the size of jumps of each $\hat{Q}^n_i$ is $n^{-1/2}$ and $\eta > n^{-1/2}$ for large enough n, there must be an arrival to that queue prior to t_2. Hence one has $0 \le t_1 \le t_2 \le T$ on the event under consideration. By the definition of t_1, a task was sent to queue i_{max} at that time, by which it follows that

$$\hat{Q}^n_{i_{max}}(t_1) \le \hat{Q}^n_{(d)}(t_1).$$

Therefore

$$\sum_{i=d+1}^{N} \big(\hat{Q}^n_{(i)}(t_1) - \hat{Q}^n_{i_{max}}(t_1)\big) \ge 0. \tag{21}$$

Since on Ω^n_a, $t_b \ge t_a = t_2$, we have $\hat{Q}^n_{(d)}(t) - \hat{Q}^n_{(1)}(t) \le \eta$ for $t < t_2$. It follows that $\hat{Q}^n_{i_{max}}(t_1) - \hat{Q}^n_{(1)}(t_1) \le \eta$, and so

$$\sum_{i=1}^{d} \big(\hat{Q}^n_{(i)}(t_1) - \hat{Q}^n_{i_{max}}(t_1)\big) \ge -(d-1)\eta. \tag{22}$$

Summing inequalities (21) and (22) yields

$$\sum_{i:i\neq i_{max}} \big(\hat{Q}^n_{i}(t_1) - \hat{Q}^n_{i_{max}}(t_1)\big) \ge -(d-1)\eta. \tag{23}$$

Next, by the definition of i_{max},

$$\sum_{i=d+1}^{N} \big(\hat{Q}^n_{i_{max}}(t_2) - \hat{Q}^n_{(i)}(t_2)\big) \ge 0,$$

and by the definition of t_a,

$$\sum_{i=1}^{d} \big(\hat{Q}^n_{i_{max}}(t_2) - \hat{Q}^n_{(i)}(t_2)\big) \ge d\eta. \tag{24}$$

Hence

$$\sum_{i:i\neq i_{max}} \big(\hat{Q}^n_{i_{max}}(t_2) - \hat{Q}^n_{i}(t_2)\big) \ge d\eta. \tag{25}$$

By the definition of t_1 and the right continuity of the sample paths, there are no routings to queue i_{max} during the time interval $J := [t_1, t_2]$, and so $\hat{Q}^n_{i_{max}}[J] \le 0$. Combining this with (23) and (25) gives

$$\sum_{i:i\neq i_{max}} \hat{Q}^n_i[J] \le -\eta. \tag{26}$$

Using the identity $\sum_{i=1}^N A^n_i = dA^n$ and again the fact that there are no routings to queue i_{max} during J, we have $\sum_{i:i\neq i_{max}} A^n_i[J] = dA^n[J]$. Combining this with (7), (26) and the fact that $\hat{Q}^n_i = n^{-1/2}Q^n_i$ gives

$$\sum_{i:i\neq i_{max}} D^n_i[J] + \sum_{i:i\neq i_{max}} C^n_i[J] - dA^n[J] \ge \eta n^{1/2}.$$

Using (12) and (14), this implies

$$\sum_{i:i\neq i_{max}} D_i^n[J]+(d-1)\sum_{i=1}^N D_i^n[J]+(d-1)N-dA^n[J]\geq \eta n^{1/2}. \tag{27}$$

Denote $J_i^n=\big[t_1-I_i^n(t_1),t_2-I_i^n(t_2)\big]$. By (6), we have $D_i^n[J]=S_i^n[J_i^n]$, hence by (8) and (27) we obtain

$$\sum_{i:i\neq i_{max}} \hat S_i^n[J_i^n]+(d-1)\sum_{i=1}^N \hat S_i^n[J_i^n]-d\hat A^n[J]+n^{-1/2}(d-1)N+y^n(t_2-t_1)\geq \eta, \tag{28}$$

where

$$\begin{aligned} y^n&=\Big[\sum_{i:i\neq i_{max}}\mu_i^n+(d-1)\sum_{i=1}^N\mu_i^n-d\lambda^n\Big]n^{-1/2}\\ &=-\mu_{i_{max}}^n n^{-1/2}+dv^n,\\ v^n&=\Big(\sum_{i=1}^N\mu_i^n-\lambda^n\Big)n^{-1/2}. \end{aligned} \tag{29}$$

By (1), (2) and (3), v^n is a bounded sequence. Moreover, for n large enough, $y^n<0$.

Fix a sequence $r_n>0$ such that $r_n\to 0$ and $n^{1/2}r_n\to\infty$. Let t_3 denote the first time after t_1 when all tasks present in the system at t_1 have either departed or been canceled. Note that, on the event $\{t_1\leq T\wedge\tau\}$, $\bar\delta_z^n$ is an upper bound on t_3-t_1. Moreover, it is impossible to have $t_3<t_2$, because one would then have that during $[t_3,t_2]$ no tasks are routed to queue i_{max}, by which $Q_{i_{max}}^n(t_2)=0$, in contradiction to (24). Therefore $t_3\geq t_2$, and so $t_2-t_1\leq\bar\delta^n$. We obtain

$$\mathbb{P}(\Omega_a^n\cap\{t_2-t_1\geq r_n\})\leq\mathbb{P}(\bar\delta_z^n\geq r_n)\to 0,\qquad \text{as } n\to\infty,$$

by Lemma 3.1.

Next, by (28) and the fact that $y^n\leq 0$ for large n,

$$\mathbb{P}(\Omega_a^n\cap\{t_2-t_1<r_n\})\leq\mathbb{P}\Big(d\sum_{i=1}^N w_T(\hat S_i^n,r_n)+dw_T(\hat A^n,r_n)\geq\eta\Big)\to 0,\qquad \text{as } n\to\infty,$$

by the C-tightness of $\{\hat S_i^n\}$ and $\{\hat A^n\}$. As a result,

$$\lim_n\mathbb{P}(\Omega_a^n)=0. \tag{30}$$

Step 2: estimating $\mathbb{P}(\Omega_b^n)$. On this event there is a difference of η within the d minimal normalized queue lengths at t_2, however the large $N-d$ ones are still η close.

Denote by i_{min} the (minimal) i for which $\hat Q_i^n(t_2)=\hat Q_{(1)}^n(t_2)$. By the definition of t_b, we have on Ω_b^n,

$$\hat Q_{(d)}^n(t_2)-\hat Q_{i_{min}}^n(t_2)\geq\eta, \tag{31}$$

and so

$$\sum_{i:i\neq i_{min}} \hat{Q}_i^n(t_2) - (N-1)\hat{Q}_{i_{min}}^n(t_2) = \sum_{i:i\neq i_{min}} \Big(\hat{Q}_i^n(t_2) - \hat{Q}_{i_{min}}^n(t_2)\Big)$$
$$\geq \sum_{i=r}^{N} \Big(\hat{Q}_{(i)}^n(t_2) - \hat{Q}_{i_{min}}^n(t_2)\Big)$$
$$\geq (N-d+1)\eta. \tag{32}$$

Define

$$t_1 = \sup\{t < t_2 : \hat{Q}_{(d)}^n(t) - \hat{Q}_{i_{min}}^n(t) < b\eta\}, \tag{33}$$

where $b = (N-1)^{-1}/2$. Recalling that the initial condition is zero, it follows that $t_1 < t_2$. Moreover, during the time interval $J = [t_1, t_2]$ we have $\hat{Q}_{i_{min}}^n < \hat{Q}_{(d)}^n$, hence every job arrival results with a task routed to queue i_{min} during this interval.

Next, recalling again that the size of jumps of the normalized queue length is $n^{-1/2}$, we have

$$\hat{Q}_{(d)}^n(t_1) - \hat{Q}_{i_{min}}^n(t_1) \leq b\eta + n^{-1/2}. \tag{34}$$

Therefore

$$\sum_{i=1}^{d} \Big(\hat{Q}_{(i)}^n(t_1) - \hat{Q}_{i_{min}}^n(t_1)\Big) \leq (d-1)b\eta + (d-1)n^{-1/2}. \tag{35}$$

Since on the event Ω_b^n we have $t_a > t_b$, it follows that

$$\hat{Q}_{(N)}^n(t_1) - \hat{Q}_{(d)}^n(t_1) \leq \eta. \tag{36}$$

Using (34) and (36),

$$\sum_{i=d+1}^{N} \Big(\hat{Q}_{(i)}^n(t_1) - \hat{Q}_{i_{min}}^n(t_1)\Big) \leq (N-d)(b+1)\eta + (N-d)n^{-1/2}. \tag{37}$$

Inequalities (35) and (37) yield

$$\sum_{i:i\neq i_{min}} \hat{Q}_i^n(t_1) - (N-1)\hat{Q}_{i_{min}}^n(t_1) = \sum_{i:i\neq i_{min}} \Big(\hat{Q}_i^n(t_1) - \hat{Q}_{i_{min}}^n(t_1)\Big)$$
$$\leq (N-d+b(N-1))\eta + (N-1)n^{-1/2}. \tag{38}$$

By combining (32), (38) and substituting the value for b we obtain

$$\sum_{i:i\neq i_{min}} \hat{Q}_i^n[J] - (N-1)\hat{Q}_{i_{min}}^n[J] \geq \frac{\eta}{2} - (N-1)n^{-1/2} \geq \frac{\eta}{4}, \tag{39}$$

provided that n is sufficiently large.

Recall that, on the event Ω_b^n, during the interval J, for every arriving job, one out of the d tasks is routed to queue i_{min}. By this and relation (7), we can write (39) as

$$ND_{i_{min}}^n[J] - \sum_{i=1}^{N} D_i^n[J] + NC_{i_{min}}^n[J] - \sum_{i=1}^{N} C_i^n[J] - (N-d)A^n[J] \geq \frac{\eta n^{1/2}}{4}.$$

Using (12) and (15) to bound from above the fourth and, respectively, third term on the LHS, and using (14), yields

$$\begin{aligned}&ND^n_{i_{min}}[J]-\sum_{i=1}^N D^n_i[J]\\&+N\sum_{i:i\neq i_{min}}D^n_i[J]+N^2-(d-1)\sum_{i=1}^N D^n_i[J]+(d-1)N-(N-d)A^n[J]\geq\frac{\eta n^{1/2}}{4}.\end{aligned}$$

Thus

$$\sum_{i=1}^N D^n_i[J]-A^n[J]+2N^2\geq\hat{\eta}n^{1/2}, \tag{40}$$

where $\hat{\eta}=\eta/[4(N-d)]$.

Similarly to Step 1 we let $J^n_i=[t_1-I^n_i(t_1),t_2-I^n_i(t_2)]$, and using (6) and (8), write (40) as

$$\sum_{i=1}^N \hat{S}^n_i[J^n_i]-\hat{A}^n[J]+n^{-1/2}2N^2+v^n(t_2-t_1)\geq\hat{\eta}, \tag{41}$$

where v^n is as in (29). Note that $n^{-1/2}2N^2\to 0$.

With r_n as before, we first estimate $\mathbb{P}(\Omega^n_b\cap\{t_2-t_1\geq r_n\})$, by a variation on the idea presented in Step 1 for bounding $\mathbb{P}(\Omega^n_a\cap\{t_2-t_1\geq r_n\})$. That is, with t_3 denoting the first time after t_1 when all tasks present in the system at t_1 are no longer in the system, we argue that it is impossible to have $t_3<t_2$. Suppose $t_3<t_2$. Again, we use the fact that during the time interval $[t_1,t_2]$, for every arriving job, one task is routed to queue i_{min}. Moreover, during $[t_3,t_2]$, all tasks present in the system correspond to arrivals between $[t_1,t_2]$. Therefore, given any time $t\in[t_3,t_2]$ and any $i\in\{1,\dots,N\}$ other than i_{min}, every task that is present in queue i at time t must have arrived after t_1 and has not yet departed or been canceled, and thus has a sibling task in queue i_{min} (corresponding to the same job). This shows that $Q_{i_{min}}(t)\geq Q_i(t)$ for every $i\in\{1,\dots,N\}$, in contradiction to (31). Therefore, on Ω^n_b, we have $t_3\geq t_2$, and so $t_2-t_1\leq t_3-t_1\leq\bar{\delta}^n_z$. Again, by Lemma 3.1, this shows

$$\lim_n\mathbb{P}(\Omega^n_b\cap\{t_2-t_1\geq r_n\})=0.$$

Next, consider the event $\Omega^n_b\cap\{t_2-t_1<r_n\}$. Recalling that the sequence v^n is bounded, say by a constant c, we have from (41),

$$\sum_i w_T(\hat{S}^n_i,r_n)+w_T(\hat{A}^n,r_n)\geq\hat{\eta}-cr_n\geq\frac{\hat{\eta}}{2}, \tag{42}$$

for all sufficiently large n. Thus, $\lim_n\mathbb{P}(\Omega^n_b\cap\{t_2-t_1<r_n\})=0$ by the C-tightness of $\{\hat{S}^n_i\}$ and $\{\hat{A}^n\}$. We thus obtain that $\lim_n\mathbb{P}(\Omega^n_b)=0$.

Combined with (30), it follows by (20) that $\lim_n\mathbb{P}(\Omega^n)=0$. Since $\eta>0$ is arbitrary, this establishes the result. $\square$

Proof of Theorem 2.1 (SSC). In view of Lemma 3.2, it suffices to show that

$$\lim_{z\to\infty}\limsup_n \mathbb{P}(\tau^n < T) = 0, \tag{43}$$

where we recall from (9) that τ^n depends on z. To this end, use (7) and (12) to write a balance equation for the total number of tasks in the system,

$$\|Q^n(t)\| = \sum_{i=1}^N Q_i^n(t) = \sum_{i=1}^N A_i^n(t) - \sum_{i=1}^N D_i^n(t) - \sum_{i=1}^N C_i^n(t) = dA^n(t) - d\sum_{i=1}^N D_i^n(t) - (d-1)M^n(t).$$

Given n and $z > 4$, if $\tau^n < T$ then there exists $\sigma^n < \tau^n$ such that $\|\hat{Q}^n(\sigma^n)\| \le z/2$ whereas $\|\hat{Q}^n(t)\| \ge z/4$ for all $t \in [\sigma^n, \tau^n]$. By the definition of τ^n, $\|\hat{Q}^n(\tau^n)\| \ge z - 1$. For z fixed, the probability that $\|\hat{Q}_i^n - \hat{Q}_j^n\|_{T\wedge\tau^n} > 1$ goes to zero with n, by Lemma 3.2. Since $z/4 > 1$ this shows that for some Ω^n with $\mathbb{P}(\Omega^n) \to 1$, on $\Omega^n \cap \{\tau^n < T\}$, one has $Q_i^n(t) > 0$ for all i and all $t \in J := [\sigma^n, \tau^n]$, with σ^n as above. Hence there is no idling during J on the indicated event. As a result,

$$d\hat{A}^n[J] - d\sum_{i=1}^N \hat{S}_i^n[J_i^n] - dv^n(\tau^n - \sigma^n) = \|\hat{Q}^n(\tau^n)\| - \|\hat{Q}^n(\sigma^n)\| \ge \frac{z}{2} - 1 > \frac{z}{4},$$

with $J_i^n := [\sigma^n - I_i^n(\sigma^n), \tau^n - I_i^n(\sigma^n)]$. Hence

$$\limsup_{z\to\infty}\limsup_n \mathbb{P}(\tau^n < T) \le \limsup_{z\to\infty}\limsup_n \mathbb{P}\Big(2d\|\hat{A}^n\|_T + 2d\sum_{i=1}^N \|\hat{S}_i^n\|_T + d|v^n|T > \frac{z}{4}\Big) = 0,$$

where the tightness of the RVs $\|\hat{S}_i^n\|_T$, $\|\hat{A}^n\|_T$ and the boundedness of v^n are used. This shows (43) and completes the proof. □

Proof of Corollary 2.2 (Diffusion limit). For the number of jobs is the system, a proof that follows the arguments of the proof of Proposition 2.5 of [3] shows convergence to an (a, b)-RBM, with $a = N\hat{m}_0 = \hat{\lambda} - \sum_i \hat{\mu}_i$ and $b = N^2\hat{\sigma}_0^2 = \lambda V_{IA(1)} + \sum \mu_i V_{Ti(1)}$. For the number of tasks one multiplies by d, and then arguing via SSC as in Proposition 2.5 of [3] for the number of tasks in each buffer, one divides by N. □

Proof of Theorem 2.3 (Delay). Recall that a^n is a fixed sequence and $\hat{\Delta}^n(t) = \hat{\Delta}^n(t, t + a^n)$. Let us also denote $\Delta^n(t) = \Delta^n(t, t + a^n)$ and $K^n(t) = K^n(t, t + a^n)$, $t \ge 0$. Recall that for $k \in \mathbb{N}$, $\alpha^n(k)$ and $\beta^n(k)$ denote the arrival and, respectively, departure time of job k. Note that the set $K^n(t)$ includes all jobs to arrive during the time interval $(t, t + a^n]$. For $t \ge 0$, let $\theta^t = \theta^{n,t} = t + a^n$. Given t, we divide the collection of all jobs that have either arrived or departed during the time interval $(t, \theta^t]$ into four categories. Namely, we set

$$K_1^n(t) = \{k : t < \alpha(k) \le \beta(k) \le \theta^t\}, \qquad K_2^n(t) = \{k : t < \alpha(k) \le \theta^t < \beta^n(k)\},$$

$$K_3^n(t) = \{k : \alpha^n(k) \le t \le \beta(k) \le \theta^t\}, \qquad K_4^n(t) = \{k : \alpha(k) \le t \le \theta^t < \beta^n(k)\}.$$

Thus

- $K_1^n(t)$ are jobs that arrived and departed during $(t, \theta^t]$.

- $K_2^n(t)$ arrived during $(t, \theta^t]$, and departed after.
- $K_3^n(t)$ arrived before $(t, \theta^t]$ and departed within.
- $K_4^n(t)$ arrived before $(t, \theta^t]$ and departed after.

Note that $K^n(t) = K_1^n(t) \cup K_2^n(t)$.

With $Q_J^n(t)$ denoting the number of jobs in the system at time t, we have

$$\int_t^{\theta^t} Q_J^n(s)ds = \sum_k \int_t^{\theta^t} 1_{\{\text{job } k \text{ is in the system at time } s\}} ds$$
$$= \sum_{k \in K_1^n(t)} \delta^n(k) + \sum_{k \in K_2^n(t)} (\theta^t - \alpha^n(k)) + \sum_{k \in K_3^n(t)} (\beta^n(k) - t) + a^n |K_4^n(t)|.$$

Thus denoting

$$\hat{Q}_{\text{ave}}^n(t) = (a^n)^{-1} \int_t^{\theta^t} \hat{Q}_J^n(s)ds,$$

$$e_2(t) = n^{1/2} |K^n(t)|^{-1} \sum_{k \in K_2^n(t)} (\beta^n(k) - \theta^t), \tag{44}$$

$$e_3(t) = -n^{1/2} |K^n(t)|^{-1} \sum_{k \in K_3^n(t)} (\beta^n(k) - t), \tag{45}$$

$$e_4(t) = -n^{1/2} |K^n(t)|^{-1} a^n |K_4^n(t)|,$$

and $e(t) = e_2(t) + e_3(t) + e_4(t)$, we can write the diffusion scale average delay as

$$\hat{\Delta}^n(t) = n^{1/2} |K^n(t)|^{-1} \sum_{k \in K_1^n(t) \cup K_2^n(t)} \delta^n(k)$$
$$= n^{1/2} |K^n(t)|^{-1} \Big(\int_t^{\theta^t} Q_J^n(s)ds + \sum_{k \in K_2^n(t)} (\beta^n(k) - \theta^t) - \sum_{k \in K_3^n(t)} (\beta^n(k) - t) - a^n |K_4^n(t)| \Big)$$
$$= na^n |K^n(t)|^{-1} \hat{Q}_{\text{ave}}^n(t) + e(t). \tag{46}$$

The remaining steps are to show that $na^n|K^n|^{-1}$ and e converge in probability to λ^{-1} and 0, respectively, and then that $\hat{Q}_{\text{ave}}^n$ has the same weak limit as $d^{-1} \sum_i \hat{Q}_i^n$.

Fix $T > 0$. Using (8),

$$(na^n)^{-1} |K^n(t)| = (na^n)^{-1} (A^n(\theta^t) - A^n(t)) = n^{-1/2} (a^n)^{-1} [\hat{A}^n(\theta^t) - \hat{A}^n(t)] + n^{-1} \lambda^n. \tag{47}$$

By the tightness of the RVs $\|\hat{A}^n\|_T$, the RHS above converges to λ in probability, uniformly in $t \in [0, T]$. Hence $na^n |K^n(t)|^{-1}$ converges to λ^{-1} in probability, uniformly in $t \in [0, T]$.

We next show that $e_i \Rightarrow 0$, $i = 2, 3, 4$. Recall the definitions of $\bar{\delta}^n$ and $\bar{\delta}_z^n$ from (10). It follows from Lemma 3.1 that for any z and any sequence $\{\hat{a}^n\}$ such that $n^{1/2} \hat{a}^n \to \infty$, $\lim_n \mathbb{P}(\bar{\delta}_z^n > \hat{a}^n) = 0$. By (43), $\limsup_n \mathbb{P}(\bar{\delta}^n > \bar{\delta}_z^n)$ can be made arbitrarily small by selecting large z. As a result,

$$\lim_n \mathbb{P}(\bar{\delta}^n > \hat{a}^n) = 0. \tag{48}$$

Bound e_2 from above by replacing all the partial delays by $\bar{\delta}^n$, to get

$$\begin{aligned}
|e_2(t)| &\leq n^{1/2}|K^n(t)|^{-1}\bar{\delta}^n|K_2^n(t)| \\
&\leq n^{1/2}|K^n(t)|^{-1}\bar{\delta}^n Q_J^n(\theta^t) \\
&\leq n|K^n(t)|^{-1}\bar{\delta}^n\|\hat{Q}^n\|_T.
\end{aligned}$$

Apply (48) with a sequence $\{\hat{a}^n\}$ for which $\hat{a}^n/a^n \to 0$. Then, on events with probability tending to 1, one has

$$|e_2(t)| \leq n|K^n(t)|^{-1}\hat{a}^n\|\hat{Q}^n\|_T = na^n|K^n(t)|^{-1}\frac{\hat{a}^n}{a^n}\|\hat{Q}^n\|_T.$$

Thus by the uniform convergence of $na^n|K^n(t)|^{-1}$ and the tightness of the RVs $\|\hat{Q}^n\|_T$, it follows that $\|e_2^n\|_T \to 0$ in probability.

The proof for e_3^n uses similar arguments.

As for e_4^n, we argue that the probability that there exists even a single type 4 job for some $t \in [0, T]$ tends to zero as $n \to \infty$. To see this, apply (48) with $\{\hat{a}^n\} = \{a^n\}$, by which

$$\mathbb{P}(\sup_{t\in[0,T]}|K_4^n(t)| > 0) \leq \mathbb{P}(\text{there exists } k \leq A^n(T) \text{ such that } \delta^n(k) > a^n) = \mathbb{P}(\bar{\delta}^n > a^n) \to 0.$$

It follows that $\|e\|_T \to 0$ in probability.

Finally, $\hat{Q}^n_{\text{ave}}$ is defined as a running average of the process $\hat{Q}^n_J$. Clearly, $\|\hat{Q}^n_{\text{ave}} - \hat{Q}^n_J\|_T \leq w_{T+1}(\hat{Q}^n_J, a^n)$. The process $\hat{Q}^n_J$ is equal to $d^{-1}\sum_i \hat{Q}^n_i$ up to an error bounded by $Nn^{-1/2}$, which accounts for the fact that when a task is being processed, its $d-1$ copies are no longer in the system. Now, by Corollary 2.2, for each i, $\hat{Q}^n_i$ converges in distribution to β^0, with β^0 a $(d\hat{\mu}_0, \sigma_1^2)$-RBM. It follows that $\hat{Q}^n_J$ converges to $d^{-1}N\beta^0$, an $(N\hat{\mu}_0, d^{-2}N^2\sigma_1^2)$-RBM. By (46) and the convergence $(na^n|K^n|^{-1}, e) \Rightarrow (\lambda^{-1}, 0)$, we divide further by λ to get a $(\check{\mu}_0, \check{\sigma}_0^2)$-RBM as the weak limit of $\hat{\Delta}^n$. □

4 Discussion and simulation results

4.1 Implementation complexity of RSQ(d)

In this section we discuss the implementation complexity of RSQ(d) compared to that of R(d) and JSQ. It is a common argument in the literature that JSQ does not scale to systems where N is large, due to (a) the time it takes to find the identity of the server with the minimal queue or (b) the memory required for keeping the state of the queue lengths or (c) the communication overhead of sampling all queues whenever a job arrives. R(d) is presented as a more scalable solution compared to JSQ. Since RSQ(d) requires the complete queue length information as JSQ, arguments (a)-(c) put the scalability of RSQ(d) into question.

Contrary to this, we first argue that (a) has a negligible impact on the implementation complexity and does not affect the scalability of either JSQ or RSQ(d). Second, regarding (b) and (c), assuming that servers communicate with the dispatcher but not with each other, we argue that if R(d) can be implemented, so can JSQ and RSQ(d). Note that a similar argument concerning the communication overhead of JSQ is provided in [15].

(a) *Finding the minimum*: finding the minimum of N numbers, even if N is very large, has a negligible computational overhead when using an appropriate data structure such as a priority queue (*e.g.,* Min heap or Fibonacci heap). Finding the minimum results in only a single operation (i.e., simply looking at the head of the priority queue). For a queue length update operation, $O(\log(N))$ operations are required in the worst case (e.g., decrease-key operation in a min-heap). Even with $n = 10^6$, just a few operations are required in the worst case per queue length update. This results in a single commodity core being able to perform tens to hundreds of millions of such updates per second, hence resulting in negligible overhead.
(b) *Memory*: R(d) needs to remember the present tasks' identity and location. The required memory size depends on how R(d) is implemented. For example, the dispatcher can store the information on all present tasks or the tasks themselves can store the information on where their copies are. Any scheme that is used to handle the tasks' identity and location information in R(d) can be used in RSQ(d). Thus, compared to R(d), JSQ and RSQ(d) can require up to an additional $O(N)$ memory to store queue length information. However, even if there are millions of servers, this translates into a few MB, which is negligible compared to the memory of a single commodity server (typically with hundreds of GB of memory in DRAM).
(c) *Communication overhead*: If one assumes that R(d) can be implemented, even if only for $d = 2$, then one assumes that for every job, the dispatcher can be notified when the first of the job's tasks reaches a server (or finished being processed by a server when applying late-cancellation). Under this assumption, JSQ can also be implemented in the following way: whenever a server begins processing a new job, it notifies the dispatcher of its queue length. Thus the dispatcher always has the current state of the queue lengths and is able to route incoming jobs to the shortest queue. The communication overhead is only 1 message per job, which does not depend on N and is the same order of magnitude as other low communication policies such as the-power-of-choice [16] and join-the-idle-queue [15]. RSQ(d) can be implemented using the same scheme: whenever a server begins processing a new job, it notifies the dispatcher of the identity of the job *and* of its queue length. As in R(d), the dispatcher then sends cancellation messages to the servers containing the copies of the job. Thus RSQ(d) requires no additional communication overhead compared to R(d).

4.2 Simulations

For simplicity, we simulate a discrete time-slotted system with three heterogeneous servers. The system is initially empty. A sequence of i.i.d. Bernoulli(λ) RVs determines at which time slots jobs arrive. Thus the arrival rate is λ. For each server i, a sequence of i.i.d. Geometric(μ_i) RVs determines the service duration (measured in time slots) of each job. The service rates are $\mu_1 = \mu$ (the *slow* server), $\mu_2 = 3\mu$ (the *intermediate* server) and $\mu_3 = 10\mu$ (the *fast* server), where μ is chosen such that $\sum_i \mu_i = 14\mu = 0.5$ (we choose $\sum_i \mu_i$ to be far from 1 to avoid the case where a job arrives in almost every time slot). We set $\lambda = 0.99 \sum_i \mu_i$, *i.e.,* a load of 99%. Note that R(2) is unstable in this setting. Indeed, under $R(2)$, at least a fraction of 1/3 of the traffic on average is routed to the pair of servers $\{1, 2\}$. Thus the rate at which these servers receive traffic is at least $\lambda/3$, or 4.62μ, which is larger than their combined service rate 4μ. We compare three algorithms, RSQ(1) (JSQ), RSQ(2) and RSQ(3) (JLW). Before running the algorithms, the simulation first determines the arrival times of jobs. Then, for each job, it calculates its service duration it would require if it is routed to any one of the three servers. Thus all three algorithms run on the same stochastic primitives.

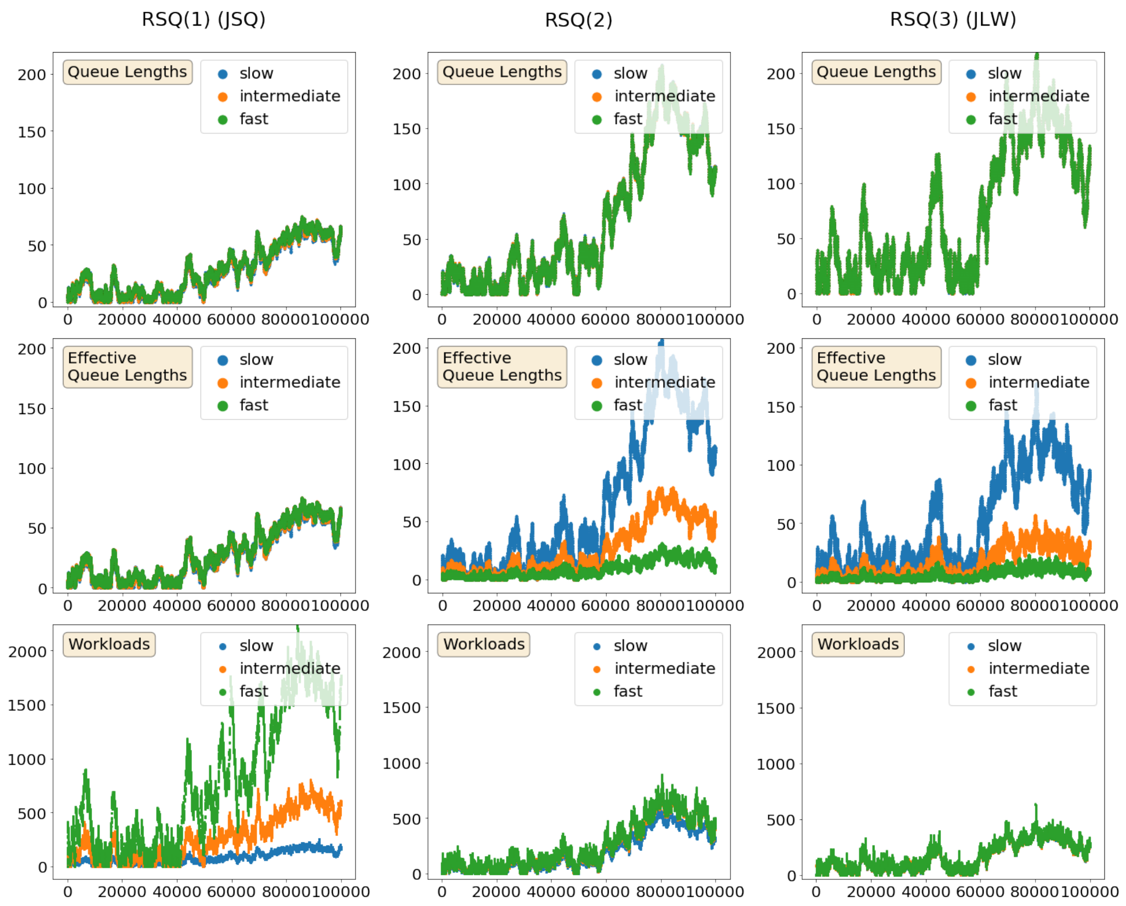

Figure 1: Queue lengths (first row), effective queue lengths (second row) and workloads (third row) under RSQ(1) (left column), RSQ(2) (middle column) and RSQ(3) (right column). As d increases, the workloads are more balanced and the total workload in the system decreases.

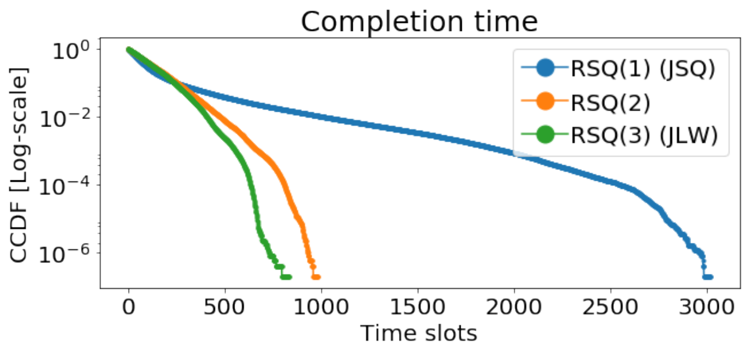

Figure 2: The Complementary Cumulative Distribution Function (in log-scale) of the completion time of jobs in a simulation with 10^7 time slots. RSQ(2) and RSQ(3) are superior to RSQ(1) with respect to tail of the distribution, such that the probability of large completion times decreases much faster for values larger than 300-400 time slots.

Our first interest lies in the *queue length*, *effective queue length* and *workload* behaviours. The queue length corresponds to the number of tasks in the buffer, including the one in service if there is any. Note that the queue length includes tasks that will be processed as well as tasks that will be cancelled. Effective queue length includes only those tasks that will be processed by the server. Note that this necessarily includes the task in service if there is any. Also, under RSQ(1), since there are no replications, the queue lengths and effective queue lengths coincide. Workload corresponds to the time it will take the server to complete all of its existing work. The workload is equal to the sum of the service duration of existing tasks that will be processed by the server, plus the remaining service duration of the task in service.

Figure 1 shows the results for 10^5 time slots. Note that queue lengths can only jump by 1 whereas workload jumps are a-priori unbounded. As expected, the queue lengths under RSQ(d) for $d \in \{1, 2, 3\}$ are balanced. This conforms with our theoretical SSC result. Also, the queue lengths increase with d, due to the increasing number of replicated tasks. As mentioned, the effective queue lengths under RSQ(1) coincide with the queue lengths and hence are balanced. However, as d increases, the effective queue lengths become less balanced. The fast server receives the largest number of jobs it will actually process, and the slow server the least. In fact, under RSQ(3), the effective queue lengths are a multiplicative factor away from one another, corresponding to their service rates (1, 3 and 10), which conforms with RSP. The opposite phenomena occurs with the workloads. Under RSQ(3) and RSQ(2) the workloads are balanced (under RSQ(2) to a lesser extent than under RSQ(3)). Under RSQ(1) the workloads are not balanced, and are a multiplicative factor away from one another corresponding to their service rates, which also conforms with RSP. The workload in the system decreases substantially as d increases from 1 to 2. This simulation thus indicates that choosing d larger than one, even if just $d = 2$, has a dramatic effect on workload balance and total workload in the system.

Next, we turn to job completion time, defined as the time a job spends in the system, from arrival until one of its tasks has finished being processed. Figure 2 displays the Complementary Cumulative Distribution Function (CCDF) of the job completion time in a simulation with 10^7 time slots. RSQ(2) and RSQ(3) are superior to RSQ(1) with respect to the tail of the distribution, such that the probability of a large completion time decreases much more rapidly after completion times around 300-400 time slots. Moreover, we observe that under RSQ(3) and RSQ(2) the completion times reach around 800 and 1000 time slots respectively, whereas under RSQ(1) they reach around 3000 time slots. The results indicate that increasing d from 1 to 2 also has a significant effect on job completion times.

Acknowledgements. The authors are grateful to an AE and two referees for careful reading and valuable comments.

References

[1] Ananthanarayanan, G., Ghodsi, A., Shenker, S., and Stoica, I. (2013). Effective straggler mitigation: Attack of the clones. *USENIX NSDI* (pp. 185-198).

[2] Ananthanarayanan, G., Kandula, S., Greenberg, A. G., Stoica, I., Lu, Y., Saha, B., and Harris, E. (2010). Reining in the outliers in map reduce clusters using Mantri. *OSDI* (Vol. 10, No. 1, p. 24).

[3] Atar, R., Keslassy, I., and Mendelson G. (2017). Sub-diffusive load-balancing in time-varying queueing systems. *Preprint.*

[4] Atar, R., and Saha, S. (2016). An ϵ-Nash equilibrium with high probability for strategic customers in heavy traffic. *Mathematics of Operations Research.*

[5] Billingsley, P. (2013). *Convergence of Probability Measures.* John Wiley and Sons.

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

[7] Chen, H., and Ye, H. Q. (2012). Asymptotic optimality of balanced routing. *Operations Research*, 60(1), 163-179.

[8] Daley, D. J. (1987). Certain optimality properties of the first-come first-served discipline for G/G/s queues. *Stochastic Processes and their Applications*, 25, 301-308.

[9] Foley, R. D., and McDonald, D. R. (2001). Join the shortest queue: stability and exact asymptotics. *Annals of Applied Probability*, 569-607.

[10] Foss, S. G. (1982). Extremal problems in queueing theory (Doctoral dissertation, PhD thesis, Novosibirsk State University. In Russian).

[11] Foss, S., and Chernova, N. (1998). On the stability of a partially accessible multi-station queue with state-dependent routing. *Queueing Systems*, 29(1), 55-73.

[12] Gardner, K., Zbarsky, S., Doroudi, S., Harchol-Balter, M., and Hyytia, E. (2015). Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(1), 347-360.

[13] Gupta, V., Balter, M. H., Sigman, K., and Whitt, W. (2007). Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation*, 64(9-12), 1062-1081.

[14] Koole, G., and Righter, R. (2008). Resource allocation in grid computing. *Journal of Scheduling*, 11(3), 163-173.

[15] Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J. R., and Greenberg, A. (2011). Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation*, 68(11), 1056-1071.

[16] Mitzenmacher, M. (2001). The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10), 1094-1104.

[17] Ousterhout, K., Wendell, P., Zaharia, M., and Stoica, I. (2013). Sparrow: distributed, low latency scheduling. *ACM SOSP* (pp. 69-84).

[18] Reiman, M. I. (1982). The heavy traffic diffusion approximation for sojourn times in Jackson networks. In *Applied probability computer science: the interface* (pp. 409-421). Birkhäuser Boston.

[19] Reiman, M. I. (1984). Some diffusion approximations with state space collapse. In *Modelling and performance evaluation methodology* (pp. 207-240). Springer Berlin Heidelberg.

[20] Shah, N. B., Lee, K., and Ramchandran, K. (2016). When do redundant requests reduce latency?. *IEEE Transactions on Communications*, 64(2), 715-722.

[21] Whitt, W. (1986). Deciding which queue to join: Some counterexamples. *Operations research*, 34(1), 55-62.

[22] Williams, R. J. (1998). Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse. *Queueing Systems*, 30(1), 27-88.

[23] Wolff, R. W. (1987). Upper bounds on work in system for multichannel queues. *Journal of applied probability*, 24(2):547–551.