The Annals of Applied Probability
2020, Vol. 30, No. 2, 870–901
https://doi.org/10.1214/19-AAP1519
© Institute of Mathematical Statistics, 2020

A LOWER BOUND ON THE QUEUEING DELAY IN RESOURCE CONSTRAINED LOAD BALANCING

BY DAVID GAMARNIK[1], JOHN N. TSITSIKLIS[2,*] AND MARTIN ZUBELDIA[2,**]

[1]*Sloan School of Management, Massachusetts Institute of Technology, gamarnik@mit.edu*
[2]*Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, *jnt@mit.edu; **mar.zubeldia@gmail.com*

We consider the following distributed service model: jobs with unit mean, general distribution, and independent processing times arrive as a renewal process of rate λn, with $0 < \lambda < 1$, and are immediately dispatched to one of several queues associated with n identical servers with unit processing rate. We assume that the dispatching decisions are made by a central dispatcher endowed with a finite memory, and with the ability to exchange messages with the servers.

We study the fundamental resource requirements (memory bits and message exchange rate), in order to drive the expected queueing delay in steady-state of a typical job to zero, as n increases. We develop a novel approach to show that, within a certain broad class of "symmetric" policies, every dispatching policy with a message rate of the order of n, and with a memory of the order of $\log n$ bits, results in an expected queueing delay which is bounded away from zero, uniformly as $n \to \infty$. This complements existing results which show that, in the absence of such limitations on the memory or the message rate, there exist policies with vanishing queueing delay (at least with Poisson arrivals and exponential service times).

1. Introduction. Distributed processing systems are ubiquitous, from passport control at the airport and checkout lines at the supermarket, to call centers and server farms for cloud computing. Many of these systems involve a stream of incoming jobs dispatched to distributed queues, with each queue associated to a different server; see Figure 1 for a stylized model. Naturally, the performance of such systems depends critically on the policy used to dispatch jobs to queues.

In order to take full advantage of the multiple servers, the dispatcher can benefit from information about the current state of the queues (e.g., whether they are empty or not). For such information to be available when a job arrives to the system and a dispatching decision is to be made, it is necessary that the dispatcher periodically obtain information about the current queue states from the corresponding servers and/or have sufficient memory that allows it to extrapolate from the available information. In this paper, we explore the tradeoff between the performance of the system and the amount of resources used to gather and maintain relevant information.

There is a variety of ways in which the system described above can be operated; these correspond to different dispatching policies, and result in different performance and resource utilization. At one extreme, the dispatcher can route incoming jobs to a queue chosen uniformly at random. This policy requires no information on the state of the queues, but the jobs experience considerable delays. At the other extreme, the dispatcher can send incoming jobs to a queue with the smallest number of jobs or to a queue with the smallest workload. The jobs in these last two policies experience little or no delay, but substantial communication overhead between the dispatcher and the servers is required.

Received July 2018; revised May 2019.
MSC2010 subject classifications. 60G99.
Key words and phrases. Load balancing, queueing delay, resource constraints.

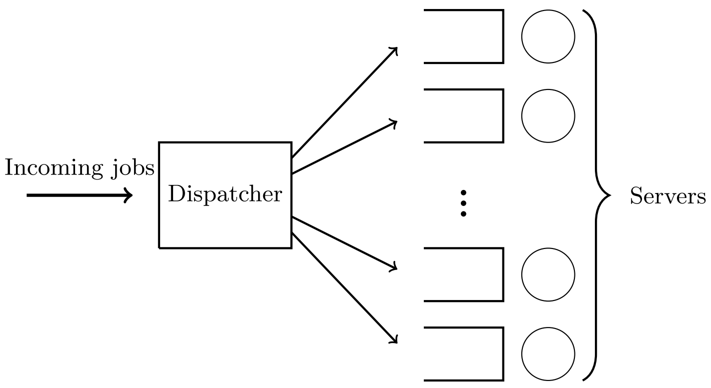

FIG. 1. *Parallel server queueing system with a central dispatcher.*

Many intermediate policies have been proposed and analyzed in the past (e.g., the power-of-d-choices or join-an-idle-queue), and they use varying amounts of resources to achieve different levels of delay performance. A more detailed discussion of the relevant literature and the various schemes therein is deferred to Section 4.

Instead of focusing on yet another policy or decision making architecture, we step back and address a more fundamental question: what is the minimum amount of resources required to obtain the best possible performance, as the number of server increases? Regarding performance, we focus on the expected time that a job has to wait before starting service. Regarding resources, we focus on the average number of messages exchanged between the dispatcher and the servers per unit of time, and on the number of bits of "long term" memory that the dispatcher has at its disposal.

1.1. *Relationship with the "balls into bins" model.* Some performance-resources tradeoffs similar to the ones we study have been analyzed in the context of the balls into bins model [2], in which n balls are to be placed sequentially into n bins. In particular, the tradeoff between the number of messages exchanged and the maximum number of balls in any one bin was recently characterized in [10], showing that a constant maximum load can be obtained using $\log n$ rounds of communication for each ball. Furthermore, the tradeoff between memory size and maximum number of balls in any one bin was studied in [1]. There, the authors showed that if the dispatcher has m bits of memory, and is only given k random choices of bins for each ball as destinations, then a constant load can only be achieved if $km \gg n$.

Note that the balls into bins model and the dynamic model that we consider are similar in that both involve sequential decisions about the destination of balls (or jobs). Furthermore, the performance metric of interest for the balls into bins model is the maximum number of balls placed in any one bin. Thus, there is the same incentive to send balls to bins with the least amount of balls in them, as in our dynamic model. As a consequence, there are many policies that are used and perform well in both settings. However, there are two fundamental differences that make the two models substantially different.

1. *Balls do not leave the system.* While this may not seem like a major difference at first glance, its impact is best highlighted by the following example: The Round-Robin policy is optimal for the balls into bins model (all bins end up with exactly one ball), but it can be far from optimal in the dynamic model.
2. *Maximum load as a performance metric.* This metric makes outliers extremely important in the balls into bins model, whereas in our dynamic model we are concerned with the

average load. Thus, a policy with a low average load and large outliers would be good for our dynamic model, but poor for the balls into bins model.

Because of these differences, results from one setting cannot be translated into the other.

1.2. *Our contribution.* We consider a broad family of decision making architectures and policies, which includes most of those considered in the earlier literature, and work toward characterizing the attainable delay performance for a given level of resources. We allow the dispatcher to have a limited memory, where it can store information on the state of the queues. We also allow the exchange of messages, either from the dispatcher to the servers (queries), or from the servers to the dispatcher (responses to queries or spontaneous status updates).

We show that if the average message rate is at most of the order of the arrival rate, and the memory size (in bits) is at most of the order of the logarithm of the number of servers, then *every* decision making architecture and policy, within a certain broad class of dispatching policies, results in a queueing delay that does *not* vanish as the system size increases. In particular, we show that the expected queueing delay in steady-state of a typical job is uniformly bounded away from zero as the number of servers goes to infinity. The main constraints that we impose on the policies that we consider are: (i) there is no queueing at the dispatcher, that is, each job is immediately dispatched to one of the parallel queues, and (ii) policies are symmetric, in a sense to be made precise later.

REMARK 1.1. For the case of Poisson arrivals and exponential service times, if either (i) the message rate is superlinear in the arrival rate [14], *or* if (ii) the memory size in bits is superlogarithmic in the number of servers and the message rate is greater than or equal to the arrival rate [7], then there exists a dispatching policy that results in a vanishing queueing delay as the system size increases. This gives sufficient conditions to achieve a vanishing queueing delay that are complementary to the necessary conditions obtained in this paper.

REMARK 1.2. The logarithmic threshold in the number of bits comes from the fact that the number of bits required to store the ID of a server is of the order of the logarithm of the number of servers. Thus, policies that have a logarithmic number of bits of memory can only store a constant number of server IDs in memory.

1.3. *Outline of the paper.* The remainder of the paper is organized as follows. In Section 2 we introduce some notation. The model and the main result are presented in Section 3. In Section 4 we discuss our result in the context of some concrete dispatching policies from the earlier literature. In Section 5 we provide the proof of our main result. Finally, in Section 6 we present our conclusions and suggestions for future work.

2. Notation.

In this section we introduce some notation that will be used throughout the paper. For any positive functions f and g, we write

$$f(n) \in \omega\big(g(n)\big) \quad \text{if and only if} \quad \liminf_{n\to\infty} \frac{f(n)}{g(n)} = \infty.$$

We let $[\cdot]^+ \triangleq \max\{\cdot, 0\}$. We let $\mathbb{Z}_+$ and $\mathbb{R}_+$ be the sets of nonnegative integers and real numbers, respectively. The indicator function is denoted by $\mathbb{1}$, so that $\mathbb{1}_A(x)$ is 1 if $x \in A$, and is 0 otherwise. Given a set A, its power set, the set of all subsets of A, is denoted by $\mathcal{P}(A)$. Random variables will always be denoted by upper case symbols. Nonrandom quantities will generally—but not always—be denoted by lower case symbols; exceptions will be pointed out as necessary.

We will use boldface fonts to denote vectors. If $\mathbf{v}$ is a vector, we denote its ith component by $\mathbf{v}_i$. We will denote the (unordered) set of elements of a vector by using the superscript "set"; for example, if $\mathbf{v} = (2, 1, 3, 1)$, then $\mathbf{v}^{\text{set}} = \{1, 2, 3\}$. Furthermore, we will use $|\mathbf{v}|$ to denote the dimension of a vector $\mathbf{v}$. If $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_m)$ is a vector, and $\mathbf{u}$ is a vector with entries in $\{1, \dots, m\}$, then $\mathbf{v}_u$ is a $|\mathbf{u}|$-dimensional vector whose ith component is $\mathbf{v}_{\mathbf{u}_i}$; for example, if $\mathbf{u} = (3, 1)$, then $\mathbf{v}_{\mathbf{u}} = (\mathbf{v}_3, \mathbf{v}_1)$.

For any positive integer n, we define the sets $\mathcal{N}_n \triangleq \{1, \dots, n\}$, and

$$\mathcal{S}_n \triangleq \left\{ \mathbf{s} \in \bigcup_{i=0}^{n} (\mathcal{N}_n)^i : \text{there are no repeated elements in } \mathbf{s} \right\},$$

where $(\mathcal{N}_n)^0 = \{\varnothing\}$. We say that a permutation $\sigma : \mathcal{N}_n \to \mathcal{N}_n$ *fixes a set* $R \subset \mathcal{N}_n$ if $\sigma(i) = i$, for all $i \in R$. Furthermore, we say that a permutation σ *preserves the ordering* of a subset $A \subset \mathcal{N}_n$ if $\sigma(i) < \sigma(j)$ whenever $i, j \in A$ and $i < j$. If $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_m)$ is a vector in $(\mathcal{N}_n)^m$ and σ is a permutation of $\mathcal{N}_n$, we denote by $\sigma(\mathbf{v})$ the vector $(\sigma(\mathbf{v}_1), \dots, \sigma(\mathbf{v}_m))$. Finally, for any function $X(\cdot)$ of time, and any $t \in \mathbb{R}$, we let $X(t^-) = \lim_{\tau \uparrow t} X(\tau)$, as long as the limit exists.

3. Model and main results.

In this section we present our main result. We first present a unified framework that defines a broad set of dispatching policies, which includes most of the policies studied in previous literature. We then present our negative result on the expected queueing delay under resource constrained policies within this set of policies.

3.1. *Modeling assumptions.*

We consider a system consisting of n parallel servers, where each server has a processing rate equal to 1. Furthermore, each server is associated with an infinite capacity FIFO queue. Jobs arrive to the system as a single renewal process of rate λn, for some fixed $\lambda < 1$. Job sizes are i.i.d., independent from the arrival process, and have an arbitrary distribution with mean 1. We use the convention that a job that is being served remains in queue until its processing is completed. We assume that each server is work-conserving: a server is idle if and only if the corresponding queue is empty.

A central controller (dispatcher) is responsible for routing each incoming job to a queue, immediately upon arrival. The dispatcher has limited information on the state of the queues; it can only rely on a limited amount of local memory and on messages that provide partial information about the state of the system. These messages (which are assumed to be instantaneous) can be sent from a server to the dispatcher at any time, or from the dispatcher to a server (in the form of queries) at the time of an arrival. Messages from a server can only contain information about the state of its own queue (number of remaining jobs and the remaining workload of each one). Within this context, a system designer has the freedom to choose a messaging policy, as well as the rules for updating the memory and for selecting the destination of an incoming job.

We are interested in the case where n is very large, in the presence of constraints on the rate of message exchanges and on the memory size. The performance metric that we focus on is the *expected queueing delay in steady-state of a typical job*, that is, the expected time between its arrival and the time at which it starts receiving service. We will formalize this definition in Section 3.3.

3.2. *Unified framework for dispatching policies.*

In this subsection we present a unified framework that describes memory-based dispatching policies. In order to do this, we introduce a sample path construction of the evolution of the system under an arbitrary policy.

Let c_n be the number of memory bits available to the dispatcher. We define the corresponding set of memory states to be $\mathcal{M}_n \triangleq \{1, \dots, 2^{c_n}\}$. Furthermore, we define the set of possible states at a server as the set of nonnegative sequences $\mathcal{Q} \triangleq \mathbb{R}_+^{\mathbb{Z}_+}$, where a sequence specifies the remaining workload of each job in that queue, including the one that is being served. (In particular, an idle server is represented by the zero sequence.) As long as a queue has a finite number of jobs, the queue state is a sequence that has only a finite number of nonzero entries. The reason that we include the workload of the jobs in the state is that we wish to allow for a broad class of policies, that can take into account the remaining workload in the queues. In particular, we allow for information-rich messages that describe the full workload sequence at the server that sends the message. We are interested in the process

$$\mathbf{Q}(\cdot) = (\mathbf{Q}_1(\cdot), \dots, \mathbf{Q}_n(\cdot)) = ((\mathbf{Q}_{1,j}(\cdot))_{j=1}^{\infty}, \dots, (\mathbf{Q}_{n,j}(\cdot))_{j=1}^{\infty}) \in \mathcal{Q}^n,$$

which describes the evolution of the workload of each job in each queue. Here $\mathbf{Q}_{i,j}(t)$ is the remaining workload of the jth job in the ith queue, at time t, which for $j \geq 2$ is simply the job's service time. We are also interested in the process $M(\cdot)$ that takes values in the set $\mathcal{M}_n$ and describes the evolution of the memory state, and in the process $Z(\cdot)$ that describes the remaining time until the next arrival of a job.

3.2.1. *Fundamental processes and initial conditions.* The processes of interest will be driven by certain common fundamental processes.

1. *Arrival process*: A delayed renewal counting process $A_n(\cdot)$ with rate λn, and event times $\{T_k\}_{k=1}^{\infty}$, defined on a probability space $(\Omega_A, \mathcal{A}_A, \mathbb{P}_A)$.
2. *Spontaneous messages process*: A Poisson counting process $R_n(\cdot)$ with rate μn, and event times $\{T_k^s\}_{k=1}^{\infty}$, defined on a probability space $(\Omega_R, \mathcal{A}_R, \mathbb{P}_R)$.
3. *Job sizes*: A sequence of i.i.d. random variables $\{W_k\}_{k=1}^{\infty}$ with mean one, defined on a probability space $(\Omega_W, \mathcal{A}_W, \mathbb{P}_W)$.
4. *Randomization variables*: Four independent and individually i.i.d. sequences of random variables $\{U_k\}_{k=1}^{\infty}$, $\{V_k\}_{k=1}^{\infty}$, $\{X_k\}_{k=1}^{\infty}$, and $\{Y_k\}_{k=1}^{\infty}$, uniform on [0, 1], defined on a common probability space $(\Omega_U, \mathcal{A}_U, \mathbb{P}_U)$.
5. *Initial conditions*: Random variables $\mathbf{Q}(0)$, $M(0)$, and $Z(0)$, defined on a common probability space $(\Omega_0, \mathcal{A}_0, \mathbb{P}_0)$.

The whole system will be defined on the associated product probability space

$$\begin{aligned}(&\Omega_A \times \Omega_R \times \Omega_W \times \Omega_U \times \Omega_0, \mathcal{A}_A \times \mathcal{A}_R \times \mathcal{A}_W \times \mathcal{A}_U \times \mathcal{A}_0,\\ &\mathbb{P}_A \times \mathbb{P}_R \times \mathbb{P}_W \times \mathbb{P}_U \times \mathbb{P}_0),\end{aligned}$$

to be denoted by $(\Omega, \mathcal{A}, \mathbb{P})$. All of the randomness in the system originates from these fundamental processes, and everything else is a deterministic function of them.

3.2.2. *A construction of sample paths.* We consider some fixed n, and provide a construction of a Markov process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$, that takes values in the set $\mathcal{Q}^n \times \mathcal{M}_n \times \mathbb{R}_+$. The memory process $M(\cdot)$ is piecewise constant, and can only jump at the time of an event. All processes to be considered will have the càdlàg property (right-continuous with left limits) either by assumption (e.g., the underlying fundamental processes) or by construction.

There are three types of events: job arrivals, spontaneous messages, and service completions. We now describe the sources of these events, and what happens when they occur.

Job arrivals: At the time of the kth event of the arrival process $A_n(\cdot)$, which occurs at time T_k and involves a job with size W_k, the following transitions happen sequentially but instantaneously.

1. First, the dispatcher chooses a vector of distinct servers $\mathbf{S}_k$, from which it solicits information about their state, according to

$$\mathbf{S}_k = f_1(M(T_k^-), W_k, U_k),$$

where $f_1 : \mathcal{M}_n \times \mathbb{R}_+ \times [0, 1] \to \mathcal{S}_n$ is a measurable function defined by the policy. Note that the set of servers that are sampled only depends on the current memory state and on the size of the incoming job, but it is chosen in a randomized way, thanks to the independent random variable U_k. Thus, we allow for randomized policies; for example, the dispatcher might choose to sample a fixed number of servers uniformly at random.

2. Then, messages are sent to the servers in the vector $\mathbf{S}_k$, and the servers respond with messages containing their queue states; thus, the information received by the dispatcher is the vector $\mathbf{Q}_{\mathbf{S}_k}$. This results in $2|\mathbf{S}_k|$ messages exchanged. Using this information, the destination of the incoming job is chosen to be

$$D_k = f_2(M(T_k^-), W_k, \mathbf{S}_k, \mathbf{Q}_{\mathbf{S}_k}(T_k^-), V_k),$$

where $f_2 : \mathcal{M}_n \times \mathbb{R}_+ \times \mathcal{S}_n \times (\bigcup_{i=0}^n \mathcal{Q}^i) \times [0, 1] \to \mathcal{N}_n$ is a measurable function defined by the policy. Note that the destination of a job can only depend on the current memory state, the job size, as well as the vector of queried servers and the state of their queues, but it is chosen in a randomized way, thanks to the independent random variable V_k. Once again, we allow for randomized policies that, for example, dispatch jobs uniformly at random.

3. Finally, the memory state is updated according to

$$M(T_k) = f_3(M(T_k^-), W_k, \mathbf{S}_k, \mathbf{Q}_{\mathbf{S}_k}(T_k^-), D_k),$$

where $f_3 : \mathcal{M}_n \times \mathbb{R}_+ \times \mathcal{S}_n \times (\bigcup_{i=0}^n \mathcal{Q}^i) \times \mathcal{N}_n \to \mathcal{M}_n$ is a measurable function defined by the policy. Note that the new memory state is obtained using the same information as for selecting the destination, plus the destination of the job, but without randomization.

Spontaneous messages: At the time of the kth event of the spontaneous message process $R_n(\cdot)$, which occurs at time T_k^s, the ith server sends a spontaneous message to the dispatcher if and only if

$$g_1(\mathbf{Q}(T_k^s), X_k) = i,$$

where $g_1 : \mathcal{Q}^n \times [0, 1] \to \{0\} \cup \mathcal{N}_n$ is a measurable function defined by the policy. In that case, the memory is updated to the new memory state

$$M(T_k^s) = g_2(M(T_k^{s-}), i, \mathbf{Q}_i(T_k^s)),$$

where $g_2 : \mathcal{M}_n \times \mathcal{N}_n \times \mathcal{Q} \to \mathcal{M}_n$ is a measurable function defined by the policy, and which prescribes the server who sends a message. On the other hand, no message is sent when $g_1(\mathbf{Q}(T_k^s), X_k) = 0$. Note that the dependence of g_1 on $\mathbf{Q}$ allows the message rate at each server to depend on the server's current workload. For example, we could let idle servers send repeated spontaneous messages (as a Poisson process) to inform the dispatcher of their idleness.

Service completions: As time progresses, the remaining workload of each job that is at the head of line in a queue decreases at a constant, unit rate. When a job's workload reaches zero, the job leaves the system and every other job advances one slot. Let $\{T_k^d(i)\}_{k=1}^\infty$ be the sequence of departure times at the ith server. At those times, the ith server sends a message to the dispatcher if and only if

$$h_1(\mathbf{Q}_i(T_k^d(i)), Y_k) = 1,$$

where $h_1 : \mathcal{Q} \times [0, 1] \to \{0, 1\}$ is a measurable function defined by the policy. In that case, the memory is updated to the new memory state

$$M(T_k^d(i)) = h_2(M(T_k^d(i)^-), i, \mathbf{Q}_i(T_k^d(i))),$$

where $h_2 : \mathcal{M}_n \times \mathcal{N}_n \times \mathcal{Q} \to \mathcal{M}_n$ is a measurable function defined by the policy. On the other hand, no message is sent when $h_1(\mathbf{Q}_i(T_k^d(i)), Y_k) = 0$.

REMARK 3.1. We have chosen to describe the collection of queried servers by a vector, implying an ordering of the servers in that collection. We could have described this collection as an (unordered) set. These two options are essentially equivalent but it turns out that the ordering provided by the vector description allows for a simpler presentation of the proof.

REMARK 3.2. For any given n, a policy is completely determined by the spontaneous message rate μ, and the functions f_1, f_2, f_3, g_1, g_2, h_1 and h_2. Furthermore, many policies in the literature that are described without explicit mention of memory or messages can be cast within our framework, as we will see in Section 4.

REMARK 3.3. The memory update functions f_3, g_2 and h_2 do not involve randomization, even though our main result could be extended in that direction. We made this choice because none of the policies introduced in earlier literature require such randomization, and because it simplifies notation and the proofs.

REMARK 3.4. We only consider the memory used to store information in between arrivals or messages. Thus, when counting the memory resources used by a policy, we do not take into account information that is used in zero time (e.g., the responses from the queries at the time of an arrival) or the memory required to evaluate the various functions that describe the policy. If that additional memory were to be accounted for, then any memory constraints would be more severe, and therefore our negative result would still hold.

The dispatching policies that we have introduced obey certain constraints:

(i) The dispatcher can only send messages to the servers at the time of an arrival, and in a single round of communication. This eliminates the possibility of policies that sequentially poll the servers uniformly at random until they find an idle one. Indeed, it can be shown that such sequential polling policies may lead to asymptotically vanishing delays, without contradicting our lower bounds. On the other hand, in practice, queries involve some processing and travel time ϵ. Were we to consider a more realistic model with $\epsilon > 0$, sequential polling would also incur positive delay.

(ii) We assume that the dispatcher must immediately send an incoming job to a server upon arrival. This prevents the dispatcher from maintaining a centralized queue and operating the system as a G/G/n queue.

We now introduce a symmetry assumption on the policies. In essence it states that at the time of a job arrival, and given the current memory state, if certain sampling and dispatching decisions and a certain memory update are possible, then a permuted version of these decisions and updates is also possible (and equally likely), starting with a suitably permuted memory state.

ASSUMPTION 3.1 (Symmetric policies). We assume that the dispatching policy is symmetric, in the following sense. For any given permutation of the servers σ, there exists a corresponding (not necessarily unique) permutation σ_M of the memory states $\mathcal{M}_n$ that satisfies all of the following properties.

1. For every $m \in \mathcal{M}_n$ and $w \in \mathbb{R}_+$, and if U is a uniform random variable on $[0, 1]$, then

$$\sigma\big(f_1(m, w, U)\big) \overset{d}{=} f_1\big(\sigma_M(m), w, U\big),$$

where $\overset{d}{=}$ stands for equality in distribution.

2. For every $m \in \mathcal{M}_n$, $w \in \mathbb{R}_+$, $\mathbf{s} \in \mathcal{S}_n$, and $\mathbf{q} \in \mathcal{Q}^{|\mathbf{s}|}$, and if V is a uniform random variable on $[0, 1]$, then[1]

$$\sigma\big(f_2(m, w, \mathbf{s}, \mathbf{q}, V)\big) \overset{d}{=} f_2\big(\sigma_M(m), w, \sigma(\mathbf{s}), \mathbf{q}, V\big).$$

3. For every $m \in \mathcal{M}_n$, $w \in \mathbb{R}_+$, $\mathbf{s} \in \mathcal{S}_n$, and $\mathbf{q} \in \mathcal{Q}^{|\mathbf{s}|}$, and $d \in \mathcal{N}_n$, we have

$$\sigma_M\big(f_3(m, w, \mathbf{s}, \mathbf{q}, d)\big) = f_3\big(\sigma_M(m), w, \sigma(\mathbf{s}), \mathbf{q}, \sigma(d)\big).$$

As a concrete illustration, our symmetry assumption implies the following. If a certain memory state mandates that the vector $(2, 4, 5)$ of servers must be sampled (with probability 1), independently from the incoming job size, then there exists some other memory state which mandates that the vector $(1, 5, 7)$ will be sampled, independently from the incoming job size, and the same holds for every 3-element vector with distinct entries. Since there are $n(n-1)(n-2)$ different vectors, there must be at least so many different memory states. This suggests that if we have too few memory states, the number of "distinguished" servers, that is, servers that are treated in a special manner is severely limited. This is a key element of the proof of the delay lower bound that we present in the next subsection.

REMARK 3.5. One may contemplate a different (stronger) definition of symmetry. For example, in the first part, we could have required that $\sigma(f_1(m, w, u)) = f_1(\sigma_M(m), w, u)$, for all $u \in [0, 1]$. While this would lead to a simpler proof, this stronger definition would be too restrictive, as explained in Appendix A.

REMARK 3.6. Note that a symmetry assumption is imposed on the memory update function f_3 at the time that a job is dispatched. However, we do not need to impose a similar assumption on the memory update functions g_2 and h_2 at the times that the dispatcher receives a message. Similarly, there is no symmetry assumption on the functions g_1 and h_1 that govern the generation of server messages. In particular, we allow each server to send spontaneous messages at its own identity-dependent, and hence asymmetric, rate.

3.3. *Delay lower bound for resource constrained policies.*

Before stating the main result of this paper, we introduce formal definitions for the average message rate between the dispatcher and the servers, and for our performance metric for the delay. Furthermore, we introduce an assumption on the arrival process.

First, given a policy of the form specified in the previous subsection, we define the *average message rate* between the dispatcher and the servers as

$$\begin{aligned}\limsup_{t\to\infty} \frac{1}{t}\Bigg[&\sum_{k=1}^{A_n(t)} 2|\mathbf{S}_k| + \sum_{k=1}^{R_n(t)} \mathbb{1}_{\mathcal{N}_n}\big(g_1(\mathbf{Q}(T_k^s), X_k)\big)\\ &+\sum_{i=1}^{n} \sum_{k:\, T_k^d(i)<t} \mathbb{1}_{\{1\}}\big(h_1(\mathbf{Q}_i(T_k^d(i)), Y_k)\big)\Bigg].\end{aligned} \tag{3.1}$$

[1]Note that the argument on the right-hand side of the relation below involves $\mathbf{q}$ rather than a permuted version of $\mathbf{q}$, even though the vector $\mathbf{s}$ gets permuted. We are essentially comparing a situation where the dispatcher queries a vector $\mathbf{s}$ and receives certain numerical values $\mathbf{q}$ with the situation where the dispatcher queries a vector $\sigma(\mathbf{s})$ and receives the *same* numerical values $\mathbf{q}$.

Second, we provide a formal definition of our performance metric for the delay. We assume that the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ is stationary, with invariant probability measure π. Since the destinations of jobs (and their queueing delays) are deterministic functions of the state and i.i.d. randomization variables, the point process of arrivals with the queueing delays as marks, is also stationary. Using this, we define the *expected queueing delay in steady-state π of a typical job*, denoted by $\mathbb{E}_\pi^0[L_0]$, as follows. If L_k is the queueing delay of the kth job under the stationary process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$, then

$$\mathbb{E}_\pi^0[L_0] \triangleq \mathbb{E}_\pi\left[\frac{1}{\lambda n t}\sum_{k=1}^{A_n(t)} L_k\right], \tag{3.2}$$

where the right-hand side is independent from t due to the stationarity of the processes involved (see [3]). Furthermore, if the stationary process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ is ergodic (in the sense that every invariant set has measure either 0 or 1 under π), we have

$$\mathbb{E}_\pi^0[L_0] = \lim_{t\to\infty}\frac{1}{A_n(t)}\sum_{k=1}^{A_n(t)} L_k \quad \text{a.s.}$$

Finally, we introduce an assumption on the arrival process.

ASSUMPTION 3.2. Let I_n be distributed as the typical inter-arrival times of the delayed renewal process $A_n(\cdot)$. We assume that there exists a constant $\overline{\epsilon} > 0$, independent from n, such that the following holds. For every $\epsilon \in (0, \overline{\epsilon}]$, there exists a positive constant δ_ϵ such that

$$\delta_\epsilon \le \mathbb{P}\left(I_n \le \frac{\epsilon}{n}\right) \le 1 - \delta_\epsilon,$$

for all n.

This assumption implies that arbitrarily small inter-arrival times of order $\Theta(1/n)$ occur with a probability that is bounded away from 0, and from 1, for all n. In particular, this excludes deterministic inter-arrival times, and inter-arrival times that can take values of order $o(1/n)$ with probability of order $1 - o(1)$. On the other hand, if $A(\cdot)$ is a delayed renewal process, where the typical inter-arrival times are continuous random variables with positive density around 0, then the process $A_n(\cdot)$, defined as $A_n(t) \triangleq A(nt)$ for all $t \ge 0$, satisfies Assumption 3.2.

We are now ready to state the main result. It asserts that within the class of symmetric policies that we consider, and under some upper bounds on the memory size (logarithmic) and the message rate (linear), the expected queueing delay in steady-state of a typical job is bounded below by a positive constant.

THEOREM 3.1 (Positive delay for resource constrained policies). *For any constants $\lambda \in (0, 1)$, $c, \alpha > 0$, and for every arrival process that satisfies Assumption 3.2, there exists a constant $\zeta(\lambda, c, \alpha) > 0$ with the following property. For any fixed n, consider a symmetric memory-based dispatching policy, that is, that satisfies Assumption 3.1, with at most $c \log_2 n$ bits of memory, with an average message rate (cf. equation (3.1)) upper bounded by αn in expectation, and under which the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ admits at least one invariant probability measure π_n. Then, for all n large enough, we have*

$$\mathbb{E}_{\pi_n}^0[L_0] \ge \zeta(\lambda, c, \alpha),$$

where $\mathbb{E}_{\pi_n}^0[L_0]$ is the expected queueing delay in steady-state π_n of a typical job.

The proof is given in Section 5.

REMARK 3.7. Theorem 3.1 only deals with memory-based dispatching policies under which $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ admits at least one invariant probability measure. This is not a substantial restriction since there are many policies with this property, for example, all policies reviewed in Section 4. On the other hand, policies without this property are not of interest.

4. Dispatching policies in the literature.

In this section we put our results in perspective by showing that various dispatching policies considered earlier in the literature are special cases of the class of symmetric dispatching policies described above. Most policies have only been studied for the case of Poisson arrivals and exponential service times, so this review is restricted to that case unless stated otherwise.

4.1. *Open-loop policies.*

4.1.1. *Random routing.* The simplest policy is to dispatch each arriving job to a random queue, with each queue being equally likely to be selected. In this case, the system behaves as n independent parallel M/M/1 queues. This policy needs no messages or memory, and has a positive queueing delay independent of n.

4.1.2. *Round Robin* (*RR*). When the dispatcher has no access to the workload of incoming jobs and no messages are allowed, it is optimal to dispatch arriving jobs to the queues in a round-robin fashion [15]. This policy does not require messages but needs $\lceil \log_2 n \rceil$ bits of memory to store the ID of the next queue to receive a job. In the limit, each queue behaves like a D/M/1 queue (see [15]). While random routing is a symmetric policy, Round Robin is not. To see this, note that a memory state i must be followed by state $i+1$, and such a transition is not permutation-invariant; in particular, the memory update function f_3 does not satisfy the symmetry assumption. Round Robin can be made symmetric by using an additional $n\lceil \log_2 n \rceil$ bits of memory to specify the order with which the different servers are selected. But in any case, this policy also has a positive queueing delay, that does not vanish as n increases.

4.2. *Policies based on queue lengths.*

4.2.1. *Join a shortest queue* (*SQ*). If we wish to minimize the queueing delay and have access to the queue lengths but not to the job sizes, an optimal policy is to have each incoming job join a shortest queue, breaking any ties uniformly at random [18]. When n goes to infinity, the queueing delay vanishes, but this policy requires a message rate of $2\lambda n^2$ (n queries and n responses for each arrival), and no memory. This policy is symmetric and achieves vanishing delay, but uses a superlinear number of messages.

4.2.2. *Join a shortest of d random queues* (*SQ*(d)). In order to sharply decrease the number of messages sent, Mitzenmacher [12] and Vvedenskaya et al. [17] introduced the power-of-d-choices policy. When there is an arrival, d servers are chosen uniformly at random, and the job is sent to a shortest queue among those d servers. This policy fits our framework, and in particular is symmetric; it uses $2\lambda dn$ messages per unit of time, and zero memory. This policy was also analyzed in the case of heavy-tailed service times by Bramson et al. [5], yielding similar results. In any case, this policy has positive delay, which is consistent with Theorem 3.1.

4.2.3. *Join a shortest of d_n random queues* ($SQ(d_n)$). More recently, Mukherjee et al. [14] analyzed a variation of the SQ(d) policy, which lets d be a function of the system size n. This policy is symmetric, uses $2\lambda d_n n$ messages per unit of time and zero memory, and has zero delay as long as $d_n \to \infty$, which is consistent with Theorem 3.1.

4.2.4. *Join a shortest of d queues, with memory* ($SQ(d, b)$). Another improvement over the power-of-d-choices, proposed by Mitzenmacher et al. in [13], is obtained by using extra memory to store the IDs of the b (with $b \leq d$) least loaded queues known at the time of the previous arrival. When a new job arrives, d queues are sampled uniformly at random and the job is sent to a least loaded queue among the d sampled and the b stored queues. This policy is symmetric, needs $2\lambda dn$ messages per unit of time and $\Omega(b \log_2 n)$ bits of memory, and has positive delay, consistent with Theorem 3.1.

4.2.5. *SQ(d) for divisible jobs.* Recently, Ying et al. [19] considered the case of jobs of size m_n (with $m_n \in \omega(1)$ and $m_n/n \to 0$) arriving as a Poisson process of rate $n\lambda/m_n$, where each job can be divided into m_n tasks with mean size 1. Then, the dispatcher samples dm_n queues and does a water-filling of those queues with the m_n tasks. In this case, the number of messages sent per unit of time is $2\lambda dn$ and no memory is used. Even though this was not mentioned in [19], this policy can be shown to drive the queueing delay to 0 if $d \geq 1/(1-\lambda)$. However, this model does not fall into our framework because it involves divisible jobs.

4.3. *Policies based on remaining workload.*

4.3.1. *Join a least loaded queue* (*LL*). An appealing policy is the one that sends incoming jobs to a queue with the least remaining workload, in which case the whole system behaves as an M/M/n queue. This policy is symmetric and achieves a vanishing delay as $n \to \infty$, but it has the same quadratic messaging requirements as SQ.

4.3.2. *Join a least loaded of d queues* ($LL(d)$). A counterpart of SQ(d) is LL(d), in which the dispatcher upon arrival chooses d queues uniformly at random and sends the job to one of those queues with the least remaining workload, breaking any ties uniformly at random. This setting was studied in [9], and it does not result in asymptotically vanishing delay, consistent with Theorem 3.1.

4.4. *Policies based on job size.* The previous policies dispatched the incoming jobs based on information about the state of the queues, obtained by dynamically exchanging messages with the servers. Such information could include the remaining workload at the different queues. On the other hand, if the dispatcher only knows the size of an incoming job (which might be difficult in practice [6]), it could use a static and memoryless policy that selects a target server based solely on the job size. Harchol–Balter et al. [8] showed that delay is minimized over all such static policies by a nonsymmetric policy that partitions the set of possible job sizes into consecutive intervals and assigns each interval to a different server. This is especially effective when the jobs have highly variable sizes (e.g., heavy-tailed), yet the resulting delay can be no better than that of an M/D/1 queue, and does not vanish as $n \to \infty$. This scheme does not require any message exchanges, and could be made symmetric by using the memory to store a list of the n intervals of job sizes corresponding to each of the n servers.

4.5. *Pull-based load balancing*.

4.5.1. *Join–idle–queue* (*JIQ*). In order to reduce the message rate, Badonnel and Burgess [4], Lu et al. [11] and Stolyar [16] propose a scheme where messages are sent from a server to the dispatcher whenever the server becomes idle, so that the dispatcher can keep track of the set of idle servers in real time. Then, an arriving job is to be sent to an empty queue (if there is one) or to a queue chosen uniformly at random (if all queues are nonempty). This policy requires at most λn messages per unit of time and exactly n bits of memory (one bit for each queue, indicating whether it is empty or not). Stolyar [16] has shown that when n goes to infinity, the average delay vanishes. This policy is symmetric. It has a vanishing delay and a linear message rate, but uses superlogarithmic memory, consistent with Theorem 3.1.

4.5.2. *Resource constrained pull-based* (*RCPB*). In order to reduce the message rate and the memory usage, Gamarnik et al. [7] propose a family of dispatching policies, similar to Join–Idle–Queue, where the dispatcher keeps a small list of up to c_n idle servers, and where messages are sent from each idle server to the dispatcher as a Poisson process of rate ν_n. Then, an arriving job is sent to an empty queue (if the dispatcher knows the ID of an idle server) or to a queue chosen uniformly at random (if the dispatcher's list is empty). This policy requires $(1-\lambda)\nu_n n$ messages per unit of time and $c_n \log_2 n$ bits of memory (the size of the list, c_n, times the number of bits required to store one ID, $\log_2 n$ bits). Gamarnik et al. [7] showed that, if we either have a high message rate regime (RCPB–HMess) with $c_n \geq 1$ and $\nu_n \to \infty$, or a high memory regime (RCPB–HMem) with $c_n \to \infty$ and $\nu_n \geq \lambda/(1-\lambda)$, the expected queueing delay vanishes as $n \to \infty$. This policy is symmetric, and it only has a vanishing delay when either the message rate is superlinear, or the memory is superlogarithmic. This is consistent with Theorem 3.1.

4.6. *Memory*, *messages and queueing delay*. We now summarize the resource requirements (memory and message rate) and the asymptotic delay of the policies reviewed in this section that fall within our framework.

Note that any one of the listed policies that achieves vanishing queueing delay (see Table 1) falls into one (or both) of the following two categories:

(a) Those requiring $\omega(n)$ message rate, namely, SQ, SQ(d_n) and LL.
(b) Those requiring $\omega(\log_2 n)$ bits of memory (JIQ, and RCPB–HMem).

Our main result effectively establishes this fundamental limitation of symmetric policies.

TABLE 1
Memory usage, average message rate, and limiting delay for various policies in the literature

Policy	Memory (bits)	Message rate	Limiting delay
Random	0	0	>0
RR [15]	$\log_2 n$	0	>0
SQ [18]	0	$2\lambda n^2$	0
SQ(d) [12]	0	$2d\lambda n$	>0
SQ(d_n) [14]	0	$\omega(n)$	0
SQ(d, b) [13]	$b \log_2 n$	$2d\lambda n$	>0
LL	0	$2\lambda n^2$	0
LL(d) [9]	0	$2d\lambda n$	>0
JIQ [16]	n	λn	0
RCPB–HMess [7]	$\log_2 n$	$\omega(n)$	0
RCPB–HMem [7]	$\omega(\log_2 n)$	λn	0

5. Proof of main result.

Let us fix some n. In the sequel, we will assume that n is large enough whenever needed for certain inequalities to hold. We fix a memory-based policy that satisfies Assumption 3.1 (symmetry), with at most n^c memory states, and which results in the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ having at least one invariant probability measure. Let us fix such an invariant probability measure π_n. We consider the process in steady-state; that is, we assume that $(\mathbf{Q}(0), M(0), Z(0))$ is distributed according to π_n. Accordingly, probabilities $\mathbb{P}(\cdot)$ and expectations $\mathbb{E}[\cdot]$ encountered in the sequel will always refer to the process in steady-state.

The high-level outline of the proof is as follows. In Section 5.1 we show that under our symmetry assumption, the dispatcher can give special treatment to at most c servers, which we call *distinguished servers*. The treatment of all other servers, is symmetric, in some appropriate sense.

In Section 5.2 we consider a sequence of bad events under which, over a certain time interval, there are $c+1$ consecutive arrivals, no service completions or messages from the servers, and all sampled servers are "busy" with a substantial workload. Then, in Section 5.3, we show that this sequence of bad events has nonnegligible probability.

In Section 5.4, we develop some further consequences of the symmetry assumption, which we use to constrain the information available to the dispatcher at the time of the $(c+1)$st arrival. Loosely speaking, the idea is that during the interval of interest, the server only has information on c distinguished servers together with (useless) information on some busy servers. This in turn implies (Section 5.5) that at least one of the first $c+1$ arrivals must be dispatched to a server on which no useful information is available, and which therefore has a nonnegligible probability of inducing a nonnegligible delay, thus completing the proof.

5.1. *Local limitations of finite memory.*

We consider the (typical) case where a relatively small number of servers ($\sqrt{n}$ or less) are sampled. We will use the symmetry assumption to show that except for a small set of distinguished servers, of size at most c, all other servers must be treated as indistinguishable.

PROPOSITION 5.1. *Let U be a uniform random variable over $[0,1]$. For all n large enough, for every memory state $m \in \mathcal{M}_n$ and every possible job size $w \in \mathbb{R}_+$, the following holds. Consider any vector of servers $\mathbf{s} \in \mathcal{S}_n$ (and its associated set of servers $\mathbf{s}^{\text{set}}$) with $|\mathbf{s}| \le \sqrt{n}$, and any integer ℓ with $|\mathbf{s}|+1 \le \ell \le n$. Consider the event $B(m, w; \mathbf{s}, \ell)$ that exactly ℓ servers are sampled and that the first $|\mathbf{s}|$ of them are the same as the vector $\mathbf{s}$, that is,*

$$B(m, w; \mathbf{s}, \ell) = \{|f_1(m, w, U)| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{f_1(m, w, U)_i = \mathbf{s}_i\},$$

and assume that the conditional probability measure

$$\mathbb{P}(\cdot \mid B(m, w; \mathbf{s}, \ell))$$

is well defined. Then, there exists a unique set $R(m, w, \mathbf{s}, \ell) \subset \mathcal{N}_n \setminus \mathbf{s}^{\text{set}}$ of minimal cardinality such that

$$\mathbb{P}(f_1(m, w, U)_{|\mathbf{s}|+1} = j \mid B(m, w; \mathbf{s}, \ell)) \tag{5.1}$$

is the same for all $j \notin R(m, w, \mathbf{s}, \ell) \cup \mathbf{s}^{\text{set}}$. Furthermore, $|R(m, w, \mathbf{s}, \ell)| \le c$.

REMARK 5.1. With some notational abuse, the measure $\mathbb{P}$ in Proposition 5.1 need not correspond to the measure $\mathbb{P}$ that describes the process. We are simply considering probabilities associated with a deterministic function of the uniform random variable U.

PROOF OF PROPOSITION 5.1. Throughout the proof, we fix a particular memory state m, job size w, vector of servers $\mathbf{s}$ with $|\mathbf{s}| \le \sqrt{n}$, and an integer ℓ in the range $|\mathbf{s}|+1 \le \ell \le n$. To simplify notation, we will suppress the dependence on w.

Consider the random vector $\mathbf{S}(m) \triangleq f_1(m, U)$. Let $\mathbf{v}$ be the vector whose components are indexed by j ranging in the set $(\mathbf{s}^{\text{set}})^c = \mathcal{N}_n \setminus \mathbf{s}^{\text{set}}$, and defined for any such j, by

$$\mathbf{v}_j = \mathbb{P}\big(\mathbf{S}(m)_{|\mathbf{s}|+1} = j \mid B(m; \mathbf{s}, \ell)\big).$$

We need to show that for j outside a "small" set, all of the components of $\mathbf{v}$ are equal. Let $z_1, \dots, z_d$ be the distinct values of $\mathbf{v}_j$, as j ranges over $(\mathbf{s}^{\text{set}})^c$, and let $A_\alpha = \{j \in (\mathbf{s}^{\text{set}})^c \mid \mathbf{v}_j = z_\alpha\}$. The sequence of sets $(A_1, \dots, A_d)$ provides a partition of $(\mathbf{s}^{\text{set}})^c$ into equivalence classes, with $\mathbf{v}_j = \mathbf{v}_{j'} = z_\alpha$, for all j, j' in the αth equivalence class A_α. Let $k_1, \dots, k_d$ be the cardinalities of the equivalence classes $A_1, \dots, A_d$. Without loss of generality, assume that k_d is a largest such cardinality. We define

$$R = \{j \in (\mathbf{s}^{\text{set}})^c \mid \mathbf{v}_j \neq \mathbf{v}_d\} = A_1 \cup \cdots \cup A_{d-1},$$

so that $R^c \cap (\mathbf{s}^{\text{set}})^c = A_d$. For every $j, j' \in A_d$, we have $\mathbf{v}_j = \mathbf{v}_{j'} = \mathbf{v}_d$, and therefore the condition (5.1) is satisfied by R. Note that by choosing $\mathbf{v}_d$ to be the most common value, we are making the cardinality of the set $R^c \cap (\mathbf{s}^{\text{set}})^c = \{j \notin \mathbf{s}^{\text{set}} \mid \mathbf{v}_j = \mathbf{v}_d\}$ as large as possible, from which it follows that the set $R \cap (\mathbf{s}^{\text{set}})^c$ is as small as possible, and therefore R, as defined, is indeed a minimal cardinality subset of $(\mathbf{s}^{\text{set}})^c$ that satisfies (5.1).

We now establish the desired upper bound on the cardinality of R. Let $\Sigma_{\mathbf{s}^{\text{set}}}$ be the set of permutations that fix the set $\mathbf{s}^{\text{set}}$. Consider an arbitrary permutation $\sigma \in \Sigma_{\mathbf{s}^{\text{set}}}$ and let σ_M be a corresponding permutation of the memory states, as defined by Assumption 3.1. We let $\mathbf{v}_{\sigma^{-1}}$ be the vector with components $(\mathbf{v}_{\sigma^{-1}})_j = \mathbf{v}_{\sigma^{-1}(j)}$, for $j \notin \mathbf{s}^{\text{set}}$. Note that as we vary σ over the set $\Sigma_{\mathbf{s}^{\text{set}}}$, $\mathbf{v}_{\sigma^{-1}}$ ranges over all possible permutations of the vector $\mathbf{v}$. We also have, for $j \notin \mathbf{s}^{\text{set}}$,

$$\begin{aligned}
&(\mathbf{v}_{\sigma^{-1}})_j \\
&\quad = \mathbf{v}_{\sigma^{-1}(j)} \\
&\quad = \mathbb{P}\big(\mathbf{S}(m)_{|\mathbf{s}|+1} = \sigma^{-1}(j) \mid B(m; \mathbf{s}, \ell)\big) \\
&\quad = \mathbb{P}\Bigg(\mathbf{S}(m)_{|\mathbf{s}|+1} = \sigma^{-1}(j) \,\Bigg|\, \big\{|\mathbf{S}(m)| = \ell\big\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\mathbf{S}(m)_i = \mathbf{s}_i\}\Bigg) \\
&\quad = \mathbb{P}\Bigg(\sigma\big(\mathbf{S}(m)_{|\mathbf{s}|+1}\big) = j \,\Bigg|\, \big\{|\mathbf{S}(m)| = \ell\big\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\mathbf{S}(m)_i = \mathbf{s}_i\}\Bigg) \\
&\quad = \mathbb{P}\Bigg(\sigma\big(\mathbf{S}(m)_{|\mathbf{s}|+1}\big) = j \,\Bigg|\, \big\{|\sigma(\mathbf{S}(m))| = \ell\big\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\sigma(\mathbf{S}(m)_i) = \sigma(\mathbf{s}_i)\}\Bigg) \\
&\quad = \mathbb{P}\Bigg(\sigma\big(\mathbf{S}(m)_{|\mathbf{s}|+1}\big) = j \,\Bigg|\, \big\{|\sigma(\mathbf{S}(m))| = \ell\big\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\sigma(\mathbf{S}(m)_i) = \mathbf{s}_i\}\Bigg) \\
&\quad = \mathbb{P}\Bigg(\mathbf{S}(\sigma_M(m))_{|\mathbf{s}|+1} = j \,\Bigg|\, \big\{|\mathbf{S}(\sigma_M(m))| = \ell\big\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\mathbf{S}(\sigma_M(m))_i = \mathbf{s}_i\}\Bigg).
\end{aligned}$$

Note that in the above expressions, the only random variables are $\mathbf{S}(m)$ and $\mathbf{S}(\sigma_M(m))$, while $\mathbf{s}$ is a fixed vector. The next to last equality above holds because σ fixes the elements in the vector $\mathbf{s}$; the last equality follows because the random variables $\sigma(\mathbf{S}(m))$ and $\mathbf{S}(\sigma_M(m))$ are identically distributed, according to Part 1 of the symmetry Assumption 3.1. The equality that was established above implies that $\sigma_M(m)$ completely determines the vector $\mathbf{v}_{\sigma^{-1}}$. As $\sigma \in \Sigma_{\mathbf{s}^{\mathrm{set}}}$ changes, $\sigma_M(m)$ can take at most n^c distinct values, due to the assumed bound on the memory size, and this leads to a bound on the number of possible permutations of the vector $\mathbf{v}$:

$$\left|\{\mathbf{v}_{\sigma^{-1}} : \sigma \in \Sigma_{\mathbf{s}^{\mathrm{set}}}\}\right| \leq n^c.$$

We now argue that since $\mathbf{v}$ has relatively few distinct permutations, most of its entries $\mathbf{v}_i$ must be equal. Recall the partition of the set $(\mathbf{s}^{\mathrm{set}})^c$ of indices into equivalence classes, of sizes $k_1, \dots, k_d$, with k_d being the largest cardinality. Note that there is a one-to-one correspondence between distinct permutations $\mathbf{v}_{\sigma^{-1}}$ of the vector $\mathbf{v}$ and distinct partitions of $(\mathbf{s}^{\mathrm{set}})^c$ into a sequence of subsets of cardinalities $k_1, \dots, k_d$, with the value z_α being taken on the αth subset. It follows that the number of different partitions of S^c into sets with the given cardinalities, which is given by the multinomial coefficient, satisfies

$$\binom{n-|\mathbf{s}|}{k_1!k_2!\cdots k_d!} = \left|\{\mathbf{v}_{\sigma^{-1}} : \sigma \in \Sigma_{\mathbf{s}^{\mathrm{set}}}\}\right| \leq n^c.$$

The number of choices of a k_d-element subset is no larger than the number of partitions. Therefore,

$$\binom{n-|\mathbf{s}|}{k_d} \leq n^c.$$

An elementary calculation (cf. Lemma B.1) implies that when n is large enough, we must have either (i) $k_d \geq n - |\mathbf{s}| - c$ or (ii) $k_d \leq c$. We argue that the second possibility cannot occur. Indeed, if $k_d \leq c$, and since k_d is the largest cardinality, it follows that $k_\alpha \leq c$ for every α. Since $k_1 + \cdots + k_d = n - |\mathbf{s}|$, we obtain that the number of classes, d, is at least $\lceil (n-|\mathbf{s}|)/c \rceil$. When dealing with d different classes, the number of possible partitions is at least $d!$; this can be seen by focusing on the least-indexed entry in each of the d classes and noting that these d entries may appear in an arbitrary order. Since $|\mathbf{s}| \leq \sqrt{n}$, we have $n - |\mathbf{s}| \geq n/2$, and putting everything together, we obtain

$$\lceil (n/2c) \rceil! \leq \left\lceil \frac{n-|\mathbf{s}|}{c} \right\rceil! \leq n^c.$$

This is clearly impossible when n is large enough, and case (ii) can therefore be eliminated. We conclude that $|A_d| = k_d \geq n - |\mathbf{s}| - c$. Since $|A_1 \cup \cdots \cup A_d| = |(\mathbf{s}^{\mathrm{set}})^c| = n - |\mathbf{s}|$, it follows that $|R| = |A_1 \cup \cdots \cup A_{d-1}| \leq c$, which is the desired cardinality bound on R.

It should be apparent that any minimal cardinality set R that satisfies (5.1) must be constructed exactly as our set A_d. Thus, nonuniqueness of the set R with the desired properties will arise if and only if there is another subset A_α, with $\alpha \neq d$, with the same maximal cardinality k_d. On the other hand, since $|\mathbf{s}| \leq \sqrt{n}$, we have $k_d \geq n - |\mathbf{s}| - c > n/2$, when n is large enough. But having two disjoint subsets, A_d and A_α, each of cardinality larger than $n/2$ is impossible, which proves uniqueness. □

Using a similar argument, we can also show that the distribution of the destination of the incoming job is uniform (or zero) outside the set of sampled servers and a set of at most c distinguished servers.

PROPOSITION 5.2. *Let V be a uniform random variable over* $[0, 1]$. *For all n large enough, for every memory state $m \in \mathcal{M}_n$, every vector of indices $\mathbf{s} \in \mathcal{S}_n$ with $|\mathbf{s}| \leq \sqrt{n}$, every* *queue vector state* $\mathbf{q} \in \mathcal{Q}^{|\mathbf{s}|}$, *and every job size* $w \in \mathbb{R}_+$, *the following holds. There exists a unique set* $R'(m, w, \mathbf{s}, \mathbf{q}) \subset \mathcal{N}_n \setminus \mathbf{s}^{\text{set}}$ *of minimal cardinality such that*

$$\mathbb{P}(f_2(m, w, \mathbf{s}, \mathbf{q}, V) = j) = \mathbb{P}(f_2(m, w, \mathbf{s}, \mathbf{q}, V) = k),$$

for all $j, k \notin R'(m, w, \mathbf{s}, \mathbf{q}) \cup \mathbf{s}^{\text{set}}$. *Furthermore,* $|R'(m, w, \mathbf{s}, \mathbf{q})| \leq c$.

PROOF. The proof is analogous to the proof of the previous proposition. We start by defining a vector $\mathbf{v}$, whose components are again indexed by j ranging in the set $\mathcal{N}_n \setminus \mathbf{s}^{\text{set}}$, by

$$\mathbf{v}_j = \mathbb{P}(f_2(m, w, \mathbf{s}, \mathbf{q}, V) = j).$$

Other than this new definition of the vector $\mathbf{v}$, the rest of the proof follows verbatim the one for Proposition 5.1. □

5.2. *A sequence of "bad" events.* In this subsection we introduce a sequence of "bad" events that we will be focusing on in order to establish a positive lower bound on the delay.

Recall that T_1^s is the time of the first event of the underlying Poisson process of rate μn that generates the spontaneous messages from the servers. Recall also that we denote by $\mathbf{Q}_{i,1}(t)$ the remaining workload of the job being serviced in server i, at time t, with $\mathbf{Q}_{i,1}(t) = 0$ if no job is present at server i. Let

$$B \triangleq \left\{ i : \sum_{j=1}^{\infty} \mathbf{Q}_{i,j}(0) \geq 2\gamma \right\},$$

which is the set of servers with at least 2γ remaining workload in their queues, and let $N_b = |B|$, where $\gamma \leq 1$ is a small positive constant, independent of n, to be specified later.

Consider the following events:

(i) the first $c + 1$ jobs after time 0 are all of size at least 2γ,

$$\mathcal{A}_w \triangleq \{W_1, \dots, W_{c+1} \geq 2\gamma\};$$

(ii) the first potential spontaneous message occurs after time γ/n, and the $(c+1)$st arrival occurs before time γ/n,

$$\mathcal{A}_a \triangleq \left\{ T_1^s > \frac{\gamma}{n} \right\} \cap \left\{ T_{c+1} < \frac{\gamma}{n} \right\};$$

(iii) there are no service completions before time γ/n,

$$\mathcal{A}_s \triangleq \left\{ \mathbf{Q}_{i,1}(0) \notin \left(0, \frac{\gamma}{n}\right), \forall i \right\};$$

(iv) there are at least γn servers that each have at least 2γ remaining workload at time zero,

$$\mathcal{A}_b \triangleq \{N_b \geq \gamma n\}.$$

For an interpretation, the event

$$\mathcal{H}_0^+ \triangleq \mathcal{A}_w \cap \mathcal{A}_a \cap \mathcal{A}_s \cap \mathcal{A}_b,$$

corresponds to an unfavorable situation for the dispatcher. This is because, at time zero, the dispatcher's memory contains possibly useful information on at most c distinguished servers (Propositions 5.1 and 5.2), and has to accommodate $c + 1$ arriving jobs by time γ/n. On the other hand, a nontrivial fraction of the servers are busy and will remain so until time γ/n (event $\mathcal{A}_b$), and it is possible that sampling will not reveal any idle servers (as long as the number of sampled servers is not too large). Thus, at least one of the jobs may end up at a busy server, resulting in positive expected delay. In what follows, we go through the just outlined sequence of unfavorable events, and then, in Section 5.3, we lower bound its probability.

Starting with $\mathcal{H}_0^+$, we define a nested sequence of events, after first introducing some more notation. For $k = 1, \dots, c+1$, let $\mathbf{S}_k$ be the random (hence denoted by an upper case symbol) vector of servers that are sampled upon the arrival of the kth job; its components are denoted by $(\mathbf{S}_k)_i$. For $i = 0, 1, \dots, |\mathbf{S}_k|$, we let

$$R_{k,i} \triangleq R\big(M(T_k^-), W_k, ((\mathbf{S}_k)_1, \dots, (\mathbf{S}_k)_{i-1}), |\mathbf{S}_k|\big)$$

be the (random) subset of servers defined in Proposition 5.1 (whenever $M(T_k^-)$, W_k, $((\mathbf{S}_k)_1, \dots, (\mathbf{S}_k)_{i-1})$, and $|\mathbf{S}_k|$ are such that the proposition applies), with the convention that $((\mathbf{S}_k)_1, \dots, (\mathbf{S}_k)_{i-1}) = \varnothing$ when $i = 1$. Otherwise, we let $R_{k,i} \triangleq \varnothing$. Furthermore, we define

$$R_k \triangleq \bigcup_{i=1}^{|\mathbf{S}_k|} R_{k,i}.$$

Moreover, let D_k be the destination of the kth job, and let

$$R'_k \triangleq R'\big(M(T_k^-), W_k, \mathbf{S}_k, \mathbf{Q}_{\mathbf{S}_k}(T_k^-)\big)$$

be the (random) subset of servers defined in Proposition 5.2 (whenever $M(T_k^-)$, W_k, $\mathbf{S}_k$ and $\mathbf{Q}_{\mathbf{S}_k}(T_k^-)$ are such that the proposition applies). Otherwise, we let $R'_k \triangleq \varnothing$. Finally, given a collection of constants $\xi_1, \dots, \xi_{c+1}$, independent of n and to be determined later, we define a nested sequence of events recursively, by

$$\begin{aligned} \mathcal{H}_k^- &\triangleq \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| \le \xi_k\}, \\ \mathcal{H}_k &\triangleq \mathcal{H}_k^- \cap \{(\mathbf{S}_k)_i \in R_{k,i} \cup B, i = 1, \dots, |\mathbf{S}_k|\}, \\ \mathcal{H}_k^+ &\triangleq \mathcal{H}_k \cap \{D_k \in \mathbf{S}_k^{\mathrm{set}} \cup R'_k \cup B\}, \end{aligned} \tag{5.2}$$

for $k = 1, \dots, c+1$.

5.3. *Lower bound on the probability of "bad" events.* In this subsection, we establish a positive lower bound, valid for all n large enough, for the probability of the event $\mathcal{H}_{c+1}^+$. In order to do this, we will obtain such uniform lower bounds for the probability of $\mathcal{H}_k^+$, for $k \ge 0$, by induction. We start with the base case.

LEMMA 5.3. *There exists a constant $\alpha_0^+ > 0$, independent of n, such that*

$$\mathbb{P}(\mathcal{H}_0^+) \ge \alpha_0^+.$$

PROOF. Note that the event $\mathcal{A}_a$ only depends on the processes of arrivals and spontaneous messages after time zero, $\mathcal{A}_w$ only depends on the i.i.d. workloads $W_1, \dots, W_{c+1}$, and $\mathcal{A}_s \cap \mathcal{A}_b$ only depends on the initial queue length vector $\mathbf{Q}(0)$. It follows that

$$\mathbb{P}(\mathcal{H}_0^+) = \mathbb{P}(\mathcal{A}_a)\mathbb{P}(\mathcal{A}_w)\mathbb{P}(\mathcal{A}_s \cap \mathcal{A}_b).$$

We will now lower bound each of these probabilities.

Note that $\mathbb{P}(\mathcal{A}_a)$ is the intersection of two independent events. The first is the event that the first arrival in a Poisson process with rate μn happens after time γ/n, or equivalently, it is the event that the first arrival of a Poisson process of rate μ happens after time γ, which has positive probability that does not depend on n. The second is the event that $c+1$ arrivals of the delayed renewal process $A_n(\cdot)$ occur before time γ/n, that is, the event that $T_{c+1} < \gamma/n$. Since the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ is stationary, the first arrival time (T_1) is distributed according to the residual time of typical inter-arrival times. In particular, if F is the cumulative distribution function of typical inter-arrival times of $A_n(t)$ (which have mean $1/\lambda n$), the well-known formula for the distribution of residual times gives

$$\begin{aligned}\mathbb{P}\left(T_1 < \frac{\gamma}{n(c+1)}\right) &= \lambda n \int_0^{\frac{\gamma}{n(c+1)}} (1-F(u))\,du \\ &= \lambda \int_0^{\frac{\gamma}{c+1}} \left(1 - F\left(\frac{v}{n}\right)\right) dv.\end{aligned}$$

Recall that Assumption 3.2 states that $1 - F(v/n) \geq \delta_v > 0$, for all $v > 0$ sufficiently small, and for all n. As a result, we have

$$\begin{aligned}\lambda \int_0^{\frac{\gamma}{c+1}} \left(1 - F\left(\frac{v}{n}\right)\right) dv &\geq \lambda \int_0^{\frac{\gamma}{c+1}} \left(1 - F\left(\frac{\gamma}{n(c+1)}\right)\right) dv \\ &= \frac{\lambda\gamma}{c+1}\delta_{\frac{\gamma}{c+1}},\end{aligned} \tag{5.3}$$

for all γ sufficiently small. On the other hand, for $k = 2, \dots, c+1$, Assumption 3.2 also implies that

$$\mathbb{P}\left(T_k - T_{k-1} \leq \frac{\gamma}{n(c+1)}\right) \geq \delta_{\frac{\gamma}{(c+1)}}.$$

Combining this with equation (5.3), and using the fact that the first arrival time and the subsequent inter-arrival times are independent, we obtain

$$\begin{aligned}\mathbb{P}\left(T_{c+1} < \frac{\gamma}{n}\right) &\geq \mathbb{P}\left(\left\{T_1 < \frac{\gamma}{n(c+1)}\right\} \cap \bigcap_{k=2}^{c+1} \left\{T_k - T_{k-1} \leq \frac{\gamma}{n(c+1)}\right\}\right) \\ &\geq \mathbb{P}\left(T_1 < \frac{\gamma}{n(c+1)}\right) \prod_{k=2}^{c+1} \mathbb{P}\left(T_k - T_{k-1} \leq \frac{\gamma}{n(c+1)}\right) \\ &\geq \frac{\lambda\gamma}{c+1}(\delta_{\frac{\gamma}{c+1}})^{c+1},\end{aligned}$$

which is a positive constant independent from n.

We also have

$$\begin{aligned}\mathbb{P}(\mathcal{A}_w) &= \prod_{i=1}^{c+1} \mathbb{P}(W_i \geq 2\gamma) \\ &= \mathbb{P}(W_i \geq 2\gamma)^{c+1},\end{aligned}$$

which is independent of n, and positive for γ small enough.

We now consider the event $\mathcal{A}_s$. If $\mathcal{A}_s^c$ holds, then there exists a server i such that $0 < \mathbf{Q}_{i,1}(0) \leq \gamma/n$, and thus we have a job departure during $(0, \frac{\gamma}{n}]$. Let X be the number of service completions during $(0, \frac{\gamma}{n}]$. The occurrence of $\mathcal{A}_s^c$ implies $X \geq 1$. Furthermore, the expected number of service completions in steady-state during any fixed interval must be equal to the expected number of arrivals, so that

$$\mathbb{P}(\mathcal{A}_s^c) \leq \mathbb{E}[X] = (n\lambda)\frac{\gamma}{n} = \lambda\gamma. \tag{5.4}$$

We now consider the event $\mathcal{A}_b$. Recall that

$$N_b = \left|\left\{i : \sum_{j=1}^{\infty} \mathbf{Q}_{i,j}(0) \geq 2\gamma\right\}\right|.$$

Let

$$N_I = \left|\left\{i : \sum_{j=1}^{\infty} \mathbf{Q}_{i,j}(0) = 0\right\}\right|$$

and

$$N_d = \left|\left\{i : 0 < \sum_{j=1}^{\infty} \mathbf{Q}_{i,j}(0) < 2\gamma\right\}\right|.$$

Then, $n = N_b + N_I + N_d$. Furthermore, all servers with $0 < \sum_{j=1}^{\infty} \mathbf{Q}_{i,j}(0) < 2\gamma$ will have a departure in $(0, 2\gamma)$. Let Y be the number of departures (service completions) during $(0, 2\gamma)$. Then, $Y \geq N_d$. We use once more that the expected number of service completions in steady-state during any fixed interval must be equal to the expected number of arrivals, to obtain

$$n\lambda 2\gamma = \mathbb{E}[Y] \geq \mathbb{E}[N_d].$$

Furthermore, by applying Little's law to the number of busy servers, in steady-state, we obtain

$$\mathbb{E}[N_I] = (1 - \lambda)n.$$

Hence

$$\mathbb{E}[N_b] = n - \mathbb{E}[N_I] - \mathbb{E}[N_d] \geq n(\lambda - 2\lambda\gamma).$$

On the other hand, we have

$$\begin{aligned}\mathbb{E}[N_b] &\leq \mathbb{P}(N_b \leq \gamma n)\gamma n + \mathbb{P}(N_b > \gamma n)n \\ &\leq \gamma n + \mathbb{P}(N_b \geq \gamma n)n \\ &= \gamma n + \mathbb{P}(\mathcal{A}_b)n.\end{aligned}$$

Combining these last two inequalities, we obtain

$$\mathbb{P}(\mathcal{A}_b) \geq \lambda - 2\lambda\gamma - \gamma. \tag{5.5}$$

Finally, using equations (5.4) and (5.5), we have

$$\begin{aligned}\mathbb{P}(\mathcal{A}_s \cap \mathcal{A}_b) &= \mathbb{P}(\mathcal{A}_b) - \mathbb{P}(\mathcal{A}_b \cap \mathcal{A}_s^c) \\ &\geq \mathbb{P}(\mathcal{A}_b) - \mathbb{P}(\mathcal{A}_s^c) \\ &\geq \lambda - 2\lambda\gamma - \gamma - \gamma\lambda,\end{aligned}$$

which is a positive constant if γ is chosen small enough. □

We now carry out the inductive step, from $k - 1$ to k, in a sequence of three lemmas. We make the induction hypothesis that there exists a positive constant α_{k-1}^+ such that $\mathbb{P}(\mathcal{H}_{k-1}^+) \geq \alpha_{k-1}^+$, and we sequentially prove that there exist positive constants α_k^-, α_k, and α_k^+ such that $\mathbb{P}(\mathcal{H}_k^-) \geq \alpha_k^-$ (Lemma 5.4), $\mathbb{P}(\mathcal{H}_k) \geq \alpha_k$ (Proposition 5.5), and $\mathbb{P}(\mathcal{H}_k^+) \geq \alpha_k^+$ (Lemma 5.7).

LEMMA 5.4. *Suppose that* $\mathbb{P}(\mathcal{H}_{k-1}^+) \geq \alpha_{k-1}^+ > 0$ *and that the constant* ξ_k *is chosen to be large enough. Then, there exists a constant* $\alpha_k^- > 0$, *such that for all* n *large enough, we have* $\mathbb{P}(\mathcal{H}_k^-) \geq \alpha_k^-$.

PROOF. First, recall our assumption that the average message rate (cf. equation (3.1)) is upper bounded by αn in expectation. Therefore,

$$\mathbb{E}\left[\limsup_{t\to\infty}\frac{1}{t}\sum_{j=1}^{A_n(t)} 2|\mathbf{S}_j|\right] \le \alpha n,$$

where $A_n(t)$ is the number of arrivals until time t. By Fatou's lemma, we also have

$$\limsup_{t\to\infty}\mathbb{E}\left[\frac{1}{t}\sum_{j=1}^{A_n(t)} 2|\mathbf{S}_j|\right] \le \alpha n.$$

Recall that the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ is stationary. Then, since the sampled vectors are a deterministic function of the state, and i.i.d. randomization variables, the point process of arrivals with the sampled vectors as marks, is also stationary. As a result, the expression

$$\mathbb{E}\left[\frac{1}{t}\sum_{j=1}^{A_n(t)} 2|\mathbf{S}_j|\right]$$

is independent from t (see equation (1.2.9) of [3]). In particular, for $t = \gamma/n$, we have that

$$\mathbb{E}\left[\frac{1}{\gamma}\sum_{j=1}^{A_n(\frac{\gamma}{n})} 2|\mathbf{S}_j|\right] \le \alpha. \tag{5.6}$$

Moreover, since $k \le c+1$, we have

$$\begin{aligned}\mathbb{E}\left[\sum_{j=1}^{A_n(\frac{\gamma}{n})} |\mathbf{S}_j|\right] &\ge \mathbb{E}\left[\sum_{j=1}^{A_n(\frac{\gamma}{n})} |\mathbf{S}_j| \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right]\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \ge c+1\right)\\ &\ge \mathbb{E}\left[|\mathbf{S}_k| \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right]\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \ge c+1\right).\end{aligned}$$

Combining this with equation (5.6), we obtain

$$\mathbb{E}\left[\frac{2}{\gamma}|\mathbf{S}_k| \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right]\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \ge c+1\right) \le \alpha.$$

This yields the upper bound

$$\mathbb{E}\left[|\mathbf{S}_k| \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right] \le \frac{\alpha\gamma}{2\mathbb{P}(A_n(\frac{\gamma}{n}) \ge c+1)}. \tag{5.7}$$

On the other hand, using the fact that $\mathcal{H}_{k-1}^+ \subset \{A_n(\gamma/n) \ge c+1\}$, we have

$$\begin{aligned}&\mathbb{P}(\mathcal{H}_k^-)\\ &\quad= \mathbb{P}(\mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| \le \xi_k\})\\ &\quad= \mathbb{P}\left(\mathcal{H}_{k-1}^+ \cap \left\{A_n\left(\frac{\gamma}{n}\right) \ge c+1\right\} \cap \{|\mathbf{S}_k| \le \xi_k\}\right)\\ &\quad= \mathbb{P}\left(\mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| \le \xi_k\} \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right)\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \ge c+1\right)\\ &\quad\ge \mathbb{P}(\mathcal{H}_{k-1}^+) - \mathbb{P}\left(|\mathbf{S}_k| > \xi_k \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \ge c+1\right)\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \ge c+1\right).\end{aligned} \tag{5.8}$$

Furthermore, for any constant $\xi_k > 0$, Markov's inequality implies

$$\mathbb{P}\left(|\mathbf{S}_k| > \xi_k \,\middle|\, A_n\left(\frac{\gamma}{n}\right) \geq c+1\right) \leq \frac{\mathbb{E}[|\mathbf{S}_k| \mid A_n(\frac{\gamma}{n}) \geq c+1]}{\xi_k},$$

which combined with equation (5.8) yields

$$\mathbb{P}(\mathcal{H}_k^-) \geq \mathbb{P}(\mathcal{H}_{k-1}^+) - \frac{\mathbb{E}[|\mathbf{S}_k| \mid A_n(\frac{\gamma}{n}) \geq c+1]}{\xi_k}\mathbb{P}\left(A_n\left(\frac{\gamma}{n}\right) \geq c+1\right).$$

Applying the inequality (5.7) to the equation above, we obtain

$$\mathbb{P}(\mathcal{H}_k^-) \geq \mathbb{P}(\mathcal{H}_{k-1}^+) - \frac{\alpha\gamma}{2\xi_k}.$$

Finally, combining this with the fact that $\mathbb{P}(\mathcal{H}_{k-1}^+) \geq \alpha_{k-1}^+ > 0$, we have that

$$\mathbb{P}(\mathcal{H}_k^-) \geq \alpha_{k-1}^+ - \frac{\alpha\gamma}{2\xi_k} \triangleq \alpha_k^-,$$

which is positive for all ξ_k large enough. $\square$

PROPOSITION 5.5. *Suppose that $\mathbb{P}(\mathcal{H}_k^-) \geq \alpha_k^-$, and that the constant ξ_k is chosen large enough. Then, there exists a constant $\alpha_k > 0$, such that for all n large enough, we have $\mathbb{P}(\mathcal{H}_k) \geq \alpha_k$.*

PROOF. Recall the definitions

$$\mathcal{H}_k = \mathcal{H}_k^- \cap \{(\mathbf{S}_k)_i \in R_{k,i} \cup B, i = 1, \ldots, |\mathbf{S}_k|\}$$

and

$$\mathcal{H}_k^- = \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| \leq \xi_k\}.$$

For $i = 1, \ldots, |\mathbf{S}_k|$, let us denote

$$H_{k,i} \triangleq \{(\mathbf{S}_k)_i \in R_{k,i} \cup B\}.$$

Then,

$$\begin{aligned}
&\mathbb{P}(\mathcal{H}_k) \\
&\quad= \mathbb{P}(\mathcal{H}_k^- \cap \{(\mathbf{S}_k)_i \in R_{k,i} \cup B, i = 1, \ldots, |\mathbf{S}_k|\}) \\
&\quad= \sum_\ell \mathbb{P}\left(\mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{i=1}^{\ell} H_{k,i}\right) \\
(5.9)\quad&\quad= \sum_\ell \mathbb{P}\left(\bigcap_{i=1}^{\ell} H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\}\right)\mathbb{P}(\mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\}) \\
&\quad= \sum_\ell \mathbb{P}(\mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\}) \prod_{i=1}^{\ell} \mathbb{P}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right),
\end{aligned}$$

where the sum is over all integers ℓ such that the conditional probabilities above are well defined. Intuitively, in the last step, we are treating the selection of the random vector $\mathbf{S}_k$ as a sequential selection of its components, which leads us to consider the product of suitable conditional probabilities. The next lemma provides a lower bound for the factors in this product.

LEMMA 5.6. *For all n large enough, we have*

$$\mathbb{P}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^{+} \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right) \geq \frac{\gamma}{2},$$

for all $\ell \leq \xi_k$ and $i \leq \ell$ such that the conditional probability above is well defined.

The idea of the proof of this lemma is that when a next component, $(\mathbf{S}_k)_i$ is chosen, it is either a "distinguished" server, in the set $R_{k,i}$, or else it is a server chosen uniformly outside the set $R_{k,i}$ (cf. Proposition 5.1), in which case it has a substantial probability of being a busy server, in the set B. Although the intuition is clear, the formal argument is rather tedious and is deferred to Appendix C.

Applying Lemma 5.6 to equation (5.9), and using the fact that $\mathbb{P}(\mathcal{H}_k^-) \geq \alpha_k^- > 0$, we obtain

$$\begin{aligned}\mathbb{P}(\mathcal{H}_k) &\geq \sum_{\ell} \mathbb{P}(\mathcal{H}_{k-1}^{+} \cap \{|\mathbf{S}_k| = \ell\})\left(\frac{\gamma}{2}\right)^{\ell} \\ &\geq \mathbb{P}(\mathcal{H}_{k-1}^{+} \cap \{|\mathbf{S}_k| \leq \xi_k\})\left(\frac{\gamma}{2}\right)^{\xi_k}, \\ &= \mathbb{P}(\mathcal{H}_k^{-})\left(\frac{\gamma}{2}\right)^{\xi_k} \\ &\geq \alpha_k^{-}\left(\frac{\gamma}{2}\right)^{\xi_k} \triangleq \alpha_k > 0,\end{aligned}$$

for all n large enough. □

LEMMA 5.7. *Suppose that $\mathbb{P}(\mathcal{H}_k) \geq \alpha_k$. Then, there exist a constant $\alpha_k^+ > 0$, such that for all n large enough, we have $\mathbb{P}(\mathcal{H}_k^+) \geq \alpha_k^+$.*

The proof is similar to the proof of Proposition 5.5 but with $\xi_k = 1$, and it is omitted. Intuitively, choosing the destination of a job has the same statistical properties as choosing one more server to sample, which brings us back to the setting of Proposition 5.5.

This concludes the induction step. It follows that there exists a constant $\alpha_{c+1}^+ > 0$, which is independent of n, and such that $\mathbb{P}(\mathcal{H}_{c+1}^+) \geq \alpha_{c+1}^+$.

5.4. *Upper bound on the number of useful distinguished servers.*

Let us provide some intuition on what comes next. The dispatcher initially may treat in a nontypical manner the servers in an initial set of at most c distinguished servers. As servers get sampled, the dispatcher acquires and possibly stores information about other servers. Ultimately, at the time of the $(c+1)$st arrival, the dispatcher may have acquired information and therefore treat in a special manner (i.e., asymmetrically) the servers in the set

$$\overline{R} \triangleq \bigcup_{k=1}^{c+1} (R_k \cup R'_k).$$

Recall that, for $k = 1, \ldots, c+1$, we have

$$R_k = \bigcup_{i=1}^{|\mathbf{S}_k|} R_{k,i},$$

where each of the sets in the union has cardinality at most c, by Proposition 5.1. Furthermore, for $k = 1, \dots, c+1$, the cardinality of R'_k is also at most c, by Proposition 5.2. It follows that

$$|\overline{R}| \le c \sum_{k=1}^{c+1} (1 + |\mathbf{S}_k|). \tag{5.10}$$

If we are to rely solely on this upper bound, the size of $\overline{R}$ can be larger than $c+1$, and it is possible in principle that the knowledge of so many "distinguished" servers (in the set $\overline{R}$) is enough for the dispatcher to identify $c+1$ idle servers to which to route the first $c+1$ jobs. On the other hand, under the event $\mathcal{H}^+_{c+1}$, all new information comes from servers that are "busy" (in the set B), and hence cannot be useful for the dispatching decisions. The next proposition states that for every sample path $\omega \in \mathcal{H}^+_{c+1}$, the set of idle (and therefore, potentially useful) servers on which information is available, namely, the set $\overline{R} \setminus B$, has cardinality of at most c.

PROPOSITION 5.8. *The event $\mathcal{H}^+_{c+1}$ implies the event $|\overline{R} \setminus B| \le c$.*

PROOF. Let us fix a realization $\omega \in \mathcal{H}^+_{c+1}$. We will upper bound the number of distinct images of the set $\overline{R} \setminus B$ under permutations of the set $\mathcal{N}_n$ of servers, which will lead to an upper bound on the cardinality of the set itself. In order to simplify notation, we will suppress the notational dependence on ω of all random variables for the rest of this proof.

We introduce a subset of the set of all possible permutations of $\mathcal{N}_n$, with this subset being rich enough to lead to the desired bound. Toward this goal, we define the set

$$F \triangleq \bigcup_{k=1}^{c+1} \left(\bigcup_{i=1}^{|\mathbf{S}_k|} \big[\{(\mathbf{S}_k)_i\} \setminus R_{k,i}\big] \cup \big[\{D_k\} \setminus \big(R'_k \cup \mathbf{S}^{\text{set}}_k\big)\big] \right).$$

This is the set of servers that were sampled, or that were chosen as the destination for a job, which were not in the distinguished sets $R_{k,i}$, or $R'_k \cup \mathbf{S}^{\text{set}}_k$, respectively.

Using our assumption $\omega \in \mathcal{H}^+_{c+1}$ and the definition of $\mathcal{H}^+_{c+1}$, we have

$$\bigcup_{k=1}^{c+1} \bigcup_{i=1}^{|\mathbf{S}_k|} \{(\mathbf{S}_k)_i\} \setminus R_{k,i} \subset B \quad \text{and} \quad \bigcup_{k=1}^{c+1} \{D_k\} \setminus \big(R'_k \cup \mathbf{S}^{\text{set}}_k\big) \subset B.$$

As a result, we have $F \subset B$, and thus

$$(\overline{R} \setminus B) \cap F = \varnothing.$$

Let Σ be the set of permutations σ of the server set $\mathcal{N}_n$ that:

(i) preserve the ordering of $\overline{R} \setminus B$ in the sense defined in Section 2,
(ii) fix the set $(\overline{R} \cap B) \cup F$ and
(iii) satisfy $\sigma(\overline{R} \setminus B) \cap (\overline{R} \setminus B) = \varnothing$.

Consider two permutations $\sigma, \tau \in \Sigma$ such that $\sigma(\overline{R} \setminus B) = \tau(\overline{R} \setminus B)$. Then, the fact that σ and τ both preserve the order of $\overline{R} \setminus B$ implies that $\sigma(i) = \tau(i)$, for all $i \in \overline{R} \setminus B$.

LEMMA 5.9. *Let $\sigma, \tau \in \Sigma$, and let σ_M and τ_M, respectively, be associated permutations of the memory states as specified in Assumption 3.1 (Symmetry). Let $m(0)$ be the initial memory state, at time 0. If $\sigma_M(m(0)) = \tau_M(m(0))$, then $\sigma(\overline{R}) = \tau(\overline{R})$.*

Loosely speaking, Lemma 5.9 asserts that for the given sample path, permutations σ, τ in Σ that lead to different sets $\overline{R}$ of distinguished servers must also lead (through σ_M and τ_M)

to different initial memory states. The proof is an elementary consequence of our symmetry assumption on the underlying dynamics. However, it is tedious and is deferred to Appendix D.

By Lemma 5.9, and for $\sigma \in \Sigma$, distinct images $\sigma(\overline{R})$ must correspond to distinct memory states $\sigma_M(m(0))$. Since the number of different memory states is upper bounded by n^c, this implies that

$$\left|\{\sigma(\overline{R}) : \sigma \in \Sigma\}\right| \le n^c.$$

Furthermore, since every $\sigma \in \Sigma$ fixes the set $\overline{R} \cap B$, we have

$$\left|\{\sigma(\overline{R} \setminus B) : \sigma \in \Sigma\}\right| = \left|\{\sigma(\overline{R}) : \sigma \in \Sigma\}\right| \le n^c. \tag{5.11}$$

Recall now that the only restrictions on the image $\sigma(\overline{R} \setminus B)$ under permutations in $\sigma \in \Sigma$ is that the set $(\overline{R} \cap B) \cup F$ is fixed, and that $\sigma(\overline{R} \setminus B) \cap (\overline{R} \setminus B) = \varnothing$. This implies that $\sigma(\overline{R} \setminus B)$ can be any set of the same cardinality within $(\overline{R} \cup F)^c$. It follows that

$$\left|\{\sigma(\overline{R} \setminus B) : \sigma \in \Sigma\}\right| \ge \binom{n - |\overline{R} \cup F|}{|\overline{R} \setminus B|}. \tag{5.12}$$

Recall also that under the event $\mathcal{H}_{c+1}^+$ we must have $|\mathbf{S}_k| \le \xi_k$, for $k = 1, \dots, c+1$. Thus $|F| \le \xi_1 + \dots + \xi_{c+1} + c + 1 \triangleq f$, and using equation (5.10), $|\overline{R}| \le c(\xi_1 + \dots + \xi_{c+1}) + c + 1 \triangleq \theta$. Combining these two upper bounds, we obtain

$$\binom{n - |\overline{R} \cup F|}{|\overline{R} \setminus B|} \ge \binom{n - (f + \theta)}{|\overline{R} \setminus B|}.$$

Combining this with equations (5.11) and (5.12), we obtain the inequality

$$n^c \ge \binom{n - (f + \theta)}{|\overline{R} \setminus B|}.$$

Finally, using the bound $|\overline{R}| \le \theta$, and applying Lemma B.1, we conclude that in order for this equation to hold for all n large enough, we must have $|\overline{R} \setminus B| \le c$. □

5.5. *Completing the proof.* We are now ready to complete the proof, by arguing that at least one of the first $c+1$ arrivals must be sent to a server that is either known to be busy or to a server on which no information is available, and therefore has positive probability of being busy.

Recall that for any fixed sample path in $\mathcal{H}_{c+1}^+$, we have (cf. equation (5.2))

$$\{D_1, \dots, D_{c+1}\} \subset B \cup \bigcup_{k=1}^{c+1} (\mathbf{S}_k^{\text{set}} \cup R_k').$$

Furthermore the event $\mathcal{H}_{c+1}^+$ implies that $(\mathbf{S}_k)_i \in R_{k,i} \cup B$, for $i = 1, \dots, |\mathbf{S}_k|$ and $k = 1, \dots, c+1$. Therefore,

$$\mathbf{S}_k^{\text{set}} \subset \bigcup_{i=1}^{|\mathbf{S}_k|} R_{k,i} \cup B = R_k \cup B,$$

for $k = 1, \dots, c+1$. It follows that

$$\begin{aligned}\{D_1, \dots, D_{c+1}\} &\subset B \cup \bigcup_{k=1}^{c+1} (\mathbf{S}_k^{\text{set}} \cup R_k') \\ &\subset B \cup \bigcup_{k=1}^{c+1} (R_k \cup R_k') \\ &= B \cup \overline{R}.\end{aligned}$$

Moreover, Proposition 5.8 states that $|\overline{R} \setminus B| \le c$. Thus, either (a) there exists k such that $D_k \in B$, or (b) $D_i \in \overline{R} \setminus B$ for $i = 1, \dots, c+1$, and hence there exists a pair k, l, with $k < l$, such that $D_k = D_l$. We will now show that in both cases, the queueing delay is at least γ.

Let L_k be the queueing delay of the kth arrival. Recall that for $i \in B$, we have $\mathbf{Q}_{i,1}(0) > 2\gamma$. Then, for case (a), with $D_k = i \in B$ we have

$$L_k = (\mathbf{Q}_{i,1}(0) - T_k)^+ \ge 2\gamma - \frac{\gamma}{n} \ge \gamma > 0.$$

On the other hand, for case (b), we have

$$L_l \ge [W_k - (T_l - T_k)]^+ \ge 2\gamma - \left(\frac{\gamma}{n} - 0\right) \ge \gamma > 0.$$

In both cases, we have

$$L_1 + \cdots + L_{c+1} \ge \gamma.$$

Since this is true for every sample path in $\mathcal{H}^+_{c+1}$, we obtain

$$\mathbb{E}[L_1 + \cdots + L_{c+1} \mid \mathcal{H}^+_{c+1}] \ge \gamma. \tag{5.13}$$

Finally, recall that the process $(\mathbf{Q}(\cdot), M(\cdot), Z(\cdot))$ is stationary, with invariant probability measure π_n. Then, setting $t = \gamma/n$ in equation (3.2), we obtain

$$\begin{aligned}
\mathbb{E}^0_{\pi_n}[L_0] &= \frac{1}{\lambda\gamma}\mathbb{E}\left[\sum_{j=1}^{A_n(\frac{\gamma}{n})} L_j\right] \\
&\ge \frac{1}{\lambda\gamma}\mathbb{E}\left[\sum_{j=1}^{A_n(\frac{\gamma}{n})} L_j \,\middle|\, \mathcal{H}^+_{c+1}\right]\mathbb{P}(\mathcal{H}^+_{c+1}) \\
&\ge \frac{1}{\lambda\gamma}\mathbb{E}[L_1 + \cdots + L_{c+1} \mid \mathcal{H}^+_{c+1}]\mathbb{P}(\mathcal{H}^+_{c+1}),
\end{aligned}$$

where the last inequality comes from the fact that $\mathcal{H}^+_{c+1} \subset \{A_n(\gamma/n) \ge c+1\}$. Combining this with equation (5.13) and the fact that $\mathbb{P}(\mathcal{H}^+_{c+1}) \ge \alpha^+_{c+1} > 0$, we obtain

$$\mathbb{E}^0_{\pi_n}[L_0] \ge \frac{\alpha^+_{c+1}}{\lambda} > 0.$$

As the constant in the lower bound does not depend on n, this completes the proof of the theorem.

6. Conclusions and future work.

We showed that when we have a limited amount of memory and a modest budget of messages per unit of time, and under a symmetry assumption, all dispatching policies result in queueing delay that is uniformly bounded away from zero. In particular, this implies that the queueing delay does not vanish as the system size increases.

Our result complements the results in [7], in which the authors showed that if we have a little more of either resource, that is, if the number of memory bits or the message rate grows faster with n, then there exists a symmetric policy that drives the queueing delay to zero as $n \to \infty$. Consequently, we now have necessary and sufficient conditions on the amount of resources available to a central dispatcher, in order to achieve a vanishing queueing delay as the system size increases.

There are several interesting directions for future research. For example:

(i) All the policies in the literature that achieve a vanishing queueing delay need a message rate at least equal to the arrival rate λn. We conjecture that this is not a necessary condition for a policy to have a vanishing queueing delay, as long as it has access to the incoming job sizes.

(ii) In light of the symmetry assumption in Theorem 3.1, an immediate open question is whether the result still holds without this assumption. Our proof relies heavily on symmetry and is hard to generalize. However, perhaps (nonsymmetric) policies that use the memory to store the beginning and the end of streaks of idle servers could achieve a vanishing queueing delay in the low memory and low message rate regime where symmetric policies cannot do it.

(iii) We have focused on a system with homogeneous servers. For the case of nonhomogeneous servers, even stability can become an issue, and there are interesting tradeoffs between the resources used and the stability region. In this setting, we expect a result similar to our lower bound for queueing delay, stating that a resource constrained policy cannot be stable for every stabilizable system.

APPENDIX A: COMPARISON WITH A MORE RESTRICTIVE SYMMETRY ASSUMPTION

In this appendix we explain why the stronger symmetry assumption

$$\sigma\big(f_1(m,w,u)\big)=f_1\big(\sigma_M(m),w,u\big)\quad \forall u\in[0,1], \tag{A.1}$$

would be unduly restrictive.

Consider a policy that samples a fixed number d of servers, uniformly at random (regardless of the memory state and of the incoming job size), and that satisfies this stronger symmetry assumption. Then, $f_1(m,w,u)$ is a vector of dimension d, for all $m\in\mathcal{M}_n$, $w\in\mathbb{R}_+$, and $u\in[0,1]$. Let σ,τ be a pair of permutations such that $\sigma(f_1(m,w,u))\neq\tau(f_1(m,w,u))$. The stronger symmetry assumption in equation (A.1) implies that there exists a pair of associated permutations σ_M, τ_M of the memory states such that

$$f_1\big(\sigma_M(m),w,u\big)=\sigma\big(f_1(m,w,u)\big)\neq\tau\big(f_1(m,w,u)\big)=f_1\big(\tau_M(m),w,u\big).$$

It follows that $\sigma_M(m)\neq\tau_M(m)$, and thus there must be at least as many memory states as the number of different vectors of dimension d with different entries. There are $\binom{n}{d}d!$ such vectors, and therefore a large memory would be required to implement such a uniform sampling policy if equation (A.1) were to be enforced.

On the other hand, the symmetry assumption that we have adopted in this paper only requires equality in distribution, and uniform sampling can be achieved with only one memory state (i.e., with no bits of memory). Indeed, since the sampling of servers is done uniformly at random, we have

$$f_1(m,w,U)\overset{d}{=}\sigma\big(f_1(m,w,U)\big),$$

for all permutations σ.

This example shows that the symmetry assumption that we have adopted can be substantially weaker (and thus easier to satisfy), and allows small-memory implementation of simple natural policies.

APPENDIX B: A COMBINATORIAL INEQUALITY

We record here an elementary fact.

LEMMA B.1. *Let us fix positive integer constants a and c. Suppose that b satisfies*

$$\binom{n-a}{b} \le n^c. \tag{B.1}$$

As long as n is large enough, we must have $b \le c$ or $b \ge n-a-c$.

PROOF. Suppose that $b = c+1$. The quantity $\binom{n-a}{c+1}$ is a polynomial in n of degree $c+1$ and therefore, when n is large, (B.1) cannot hold. In the range $c+1 \le b \le (n-a)/2$, the quantity $\binom{n-a}{b}$ increases with b, and hence (B.1) cannot hold either. Using the symmetry of the binomial coefficient, a similar argument is used to exclude the possibility that $(n-a)/2 \le b \le n-a-c-1$. $\square$

APPENDIX C: PROOF OF LEMMA 5.6

In order to simplify notation, we introduce the following. For any $m \in \mathcal{M}_n$, $w \in \mathbb{R}_+$, and $b \in \mathcal{P}(\mathcal{N}_n)$, we define the event

$$\mathcal{A}_{m,w,b} \triangleq \{M(T_k^-) = m, B = b, W_k = w\},$$

and we let $\mathbb{P}_{m,w,b}$ be the conditional probability measure

$$\mathbb{P}_{m,w,b}(\cdot) \triangleq \mathbb{P}(\cdot \mid \mathcal{A}_{m,w,b}).$$

Let us fix some $\ell \le \xi_k$ and some $i \le \ell$. We have

$$\begin{aligned}
&\mathbb{P}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right) \\
&\quad = \int_{m,w,b} \mathbb{P}_{m,w,b}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right) \\
&\qquad \cdot \mathrm{d}\mathbb{P}\left(\mathcal{A}_{m,w,b} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right).
\end{aligned}$$

Moreover,

$$\begin{aligned}
&\mathbb{P}_{m,w,b}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right) \\
&\quad = \sum_{\mathbf{s}} \mathbb{P}_{m,w,b}\left(H_{k,i} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} \{(\mathbf{S}_k)_j = \mathbf{s}_j\}\right) \\
&\qquad \cdot \mathbb{P}_{m,w,b}\left(\bigcap_{j=1}^{i-1} \{(\mathbf{S}_k)_j = \mathbf{s}_j\} \,\middle|\, \mathcal{H}_{k-1}^+ \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1} H_{k,j}\right),
\end{aligned}$$

where the sum is over all $(i-1)$-dimensional vectors $\mathbf{s}$ whose components are distinct indices of servers, and such that the conditional probabilities above are well defined.

It is not hard to see that the desired result follows immediately once we establish the following claim.

CLAIM C.1. *For all n large enough, we have*

$$\mathbb{P}_{m,w,b}\left(H_{k,i} \,\middle|\, \mathcal{H}^{+}_{k-1} \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1}\{(\mathbf{S}_k)_j = \mathbf{s}_j\}\right) \geq \frac{\gamma}{2},$$

for all $(m, w, b, \mathbf{s})$ such that the conditional probability above is well defined.

PROOF. Let us fix some $(m, w, b, \mathbf{s})$. Since $\mathcal{H}^{+}_{k-1}$ implies $|B| \geq \gamma n$, we have

$$|b| \geq \gamma n. \tag{C.1}$$

On the other hand, recall that

$$H_{k,i} = \{(\mathbf{S}_k)_i \in R_{k,i} \cup B\},$$

where

$$\mathbf{S}_k = f_1(M(T_k^-), W_k, U_k),$$

and $R_{k,i}$ is equal to the set

$$R(M(T_k^-), W_k, ((\mathbf{S}_k)_1, \dots, (\mathbf{S}_k)_{i-1}), |\mathbf{S}_k|)$$

defined in Proposition 5.1, whenever the proposition applies. Otherwise, we have $R_{k,j} = \varnothing$. In any case, $R_{k,j}$ is a deterministic function of the same random variables. Then, conditioned on $M(T_k^-) = m$, $W_k = w$, $B = b$, $((\mathbf{S}_k)_1, \dots, (\mathbf{S}_k)_{j-1}) = \mathbf{s}$, and $|\mathbf{S}_k| = \ell$, we have

$$H_{k,i} = \{(f_1(m, w, U_k))_i \in r_{k,i} \cup b\},$$

where $r_{k,i}$ denotes the corresponding realization of the random set $R_{k,i}$. Note that the only randomness left in this event comes from U_k, which is a randomization random variable that is chosen independent from all the events prior to time T_k^-. It follows that $H_{k,i}$ is conditionally independent from $\mathcal{H}^{+}_{k-1}$, and thus

$$\begin{aligned}\mathbb{P}_{m,w,b}&\left(H_{k,i} \,\middle|\, \mathcal{H}^{+}_{k-1} \cap \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1}\{(\mathbf{S}_k)_j = \mathbf{s}_j\}\right)\\ &= \mathbb{P}_{m,w,b}\left(H_{k,i} \,\middle|\, \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1}\{(\mathbf{S}_k)_j = \mathbf{s}_j\}\right).\end{aligned}$$

We now define the event $G_{k,\mathbf{s},i,\ell}$ to be

$$G_{k,\mathbf{s},i,\ell} \triangleq \{|\mathbf{S}_k| = \ell\} \cap \bigcap_{j=1}^{i-1}\{(\mathbf{S}_k)_j = \mathbf{s}_j\}.$$

We are interested in bounding $\mathbb{P}_{m,w,b}(H_{k,i} \mid G_{k,\mathbf{s},i,\ell})$, which we decompose into two terms:

$$\begin{aligned}\mathbb{P}_{m,w,b}(H_{k,i} \mid G_{k,\mathbf{s},i,\ell}) &= \mathbb{P}_{m,w,b}((\mathbf{S}_k)_i \in r_{k,i} \cup b \mid G_{k,\mathbf{s},i,\ell})\\ &= \mathbb{P}_{m,w,b}((\mathbf{S}_k)_i \in r_{k,i} \mid G_{k,\mathbf{s},i,\ell})\\ &\quad + \mathbb{P}_{m,w,b}((\mathbf{S}_k)_i \in b \setminus r_{k,i} \mid G_{k,\mathbf{s},i,\ell}).\end{aligned} \tag{C.2}$$

Since the conditional probability measure $\mathbb{P}_{m,w,b}(\cdot \mid G_{k,\mathbf{s},i,\ell})$ is well defined, and since $\ell \leq \xi_k$ and $\xi_l \leq \sqrt{n}$ for all n large enough, Proposition 5.1 applies and yields

$$\mathbb{P}_{m,w,b}((\mathbf{S}_k)_i = s \mid G_{k,\mathbf{s},i,\ell}) = \mathbb{P}_{m,w,b}((\mathbf{S}_k)_i = s' \mid G_{k,\mathbf{s},i,\ell}),$$

for all $s, s' \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\}$. As a result,

$$\begin{aligned}
&\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \in b \setminus r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big)\\
&\quad \geq \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \in b \setminus \big(r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\}\big) \mid G_{k,\mathbf{s},i,\ell}\big)\\
&\quad = \frac{|b \setminus (r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\})|}{n - |r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\}|}\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\} \mid G_{k,\mathbf{s},i,\ell}\big).
\end{aligned}$$

Moreover, using the facts that $|b| \geq \gamma n$ (equation (C.1)), $|r_{k,i}| \leq c$ (Proposition 5.1), and $i \leq \ell$, we obtain

$$\begin{aligned}
&\frac{|b \setminus (r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\})|}{n - |r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\}|} \cdot \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\} \mid G_{k,\mathbf{s},i,\ell}\big)\\
&\quad \geq \frac{\gamma n - c - \ell}{n} \cdot \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\} \mid G_{k,\mathbf{s},i,\ell}\big)\\
&\quad \geq \frac{\gamma}{2} \cdot \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\} \mid G_{k,\mathbf{s},i,\ell}\big),
\end{aligned}$$

when n is large enough. Finally, since the elements of the vector $\mathbf{S}_k$ are distinct,

$$\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}\} \mid G_{k,\mathbf{s},i,\ell}\big) = \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big),$$

and therefore

$$\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \in b \setminus r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big) \geq \frac{\gamma}{2}\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big).$$

We now substitute into equation (C.2), and obtain

$$\begin{aligned}
&\mathbb{P}_{m,w,b}(H_{k,i} \mid G_{k,\mathbf{s},i,\ell})\\
&\quad \geq \mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \in r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big) + \frac{\gamma}{2}\mathbb{P}_{m,w,b}\big((\mathbf{S}_k)_i \notin r_{k,i} \mid G_{k,\mathbf{s},i,\ell}\big)\\
&\quad \geq \frac{\gamma}{2},
\end{aligned}$$

for all n large enough. □

APPENDIX D: PROOF OF LEMMA 5.9

We first prove a claim about the set-valued functions R and R' introduced in Propositions 5.1 and 5.2, respectively.

CLAIM D.1. *For every $m \in \mathcal{M}_n$, $w \in \mathbb{R}_+$, $\mathbf{s} \in \mathcal{S}_n$ with $|\mathbf{s}| \leq \sqrt{n}$, $\mathbf{q} \in \mathcal{Q}^{|\mathbf{s}|}$, and for $\ell = |\mathbf{s}| + 1, \ldots, n$, and for every permutation σ, we have $R(\sigma_M(m), w, \sigma(\mathbf{s}), \ell) = \sigma(R(m, w, \mathbf{s}, \ell))$ and $R'(\sigma_M(m), w, \sigma(\mathbf{s}), \mathbf{q}) = \sigma(R'(m, w, \mathbf{s}, \mathbf{q}))$.*

PROOF. In order to simplify notation, we suppress the dependence on w of the functions R, R', and f_1 throughout the proof of the lemma.

Let U be a uniform random variable over [0, 1]. For every $m \in \mathcal{M}_n$, we define the random vector $\mathbf{S}(m) = f_1(m, U)$. Recall that $R(m, \mathbf{s}, \ell) \subset \mathcal{N}_n \setminus \mathbf{s}^{\text{set}}$ is the unique set of minimal cardinality such that

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{S}(m)_{|\mathbf{s}|+1} = j \;\middle|\; \{|\mathbf{S}(m)| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|}\{\mathbf{S}(m)_i = \mathbf{s}_i\}\right)\\
&\quad = \mathbb{P}\left(\mathbf{S}(m)_{|\mathbf{s}|+1} = j' \;\middle|\; \{|\mathbf{S}(m)| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|}\{\mathbf{S}(m)_i = \mathbf{s}_i\}\right),
\end{aligned}$$

for all $j, j' \notin R(m, \mathbf{s}, \ell) \cup \mathbf{s}^{\text{set}}$. It is not hard to see, for example, by replacing j, j' in the above equality by $\sigma^{-1}(j), \sigma^{-1}(j') \notin R(m, \mathbf{s}, \ell) \cup \mathbf{s}^{\text{set}}$, that $\sigma(R(m, \mathbf{s}, \ell)) \subset \mathcal{N}_n \setminus \sigma(\mathbf{s}^{\text{set}})$ is the unique set of minimal cardinality such that

$$\mathbb{P}\left(\sigma(\mathbf{S}(m)_{|\mathbf{s}|+1}) = j \;\middle|\; \{|\sigma(\mathbf{S}(m))| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\sigma(\mathbf{S}(m)_i) = \sigma(\mathbf{s}_i)\}\right)$$
$$= \mathbb{P}\left(\sigma(\mathbf{S}(m)_{|\mathbf{s}|+1}) = j' \;\middle|\; \{|\sigma(\mathbf{S}(m))| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\sigma(\mathbf{S}(m)_i) = \sigma(\mathbf{s}_i)\}\right),$$

for all $j, j' \notin \sigma(R(m, \mathbf{s}, \ell)) \cup \sigma(\mathbf{s}^{\text{set}})$. On the other hand, the symmetry assumption states that

$$\sigma(\mathbf{S}(m)) \overset{d}{=} \mathbf{S}(\sigma_M(m)).$$

Combining the last two equalities we get that $\sigma(R(m, \mathbf{s}, \ell)) \subset \mathcal{N}_n \setminus \sigma(\mathbf{s}^{\text{set}})$ is the unique set of minimal cardinality such that

$$\mathbb{P}\left(\mathbf{S}(\sigma_M(m))_{|\mathbf{s}|+1} = j \;\middle|\; \{|\mathbf{S}(\sigma_M(m))| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\mathbf{S}(\sigma_M(m))_i = \sigma(\mathbf{s}_i)\}\right)$$
$$= \mathbb{P}\left(\mathbf{S}(\sigma_M(m))_{|\mathbf{s}|+1} = j' \;\middle|\; \{|\mathbf{S}(\sigma_M(m))| = \ell\} \cap \bigcap_{i=1}^{|\mathbf{s}|} \{\mathbf{S}(\sigma_M(m))_i = \sigma(\mathbf{s}_i)\}\right),$$

for all $i, j \notin \sigma(R(m, \mathbf{s}, \ell)) \cup \sigma(\mathbf{s}^{\text{set}})$. However, this is exactly the definition of $R(\sigma_M(m), \sigma(\mathbf{s}), \ell)$ (uniqueness is crucial at this point), so we have

$$\sigma(R(m, \mathbf{s}, \ell)) = R(\sigma_M(m), \sigma(\mathbf{s}), \ell).$$

The proof of $R'(\sigma_M(m), \sigma(\mathbf{s}), \mathbf{q}) = \sigma(R'(m, \mathbf{s}, \mathbf{q}))$ is analogous (this time making use of the symmetry of the mapping f_2) and is omitted. $\square$

We continue with the proof of Lemma 5.9. Under the event $\mathcal{H}^+_{c+1}$, we have $(\mathbf{S}_1)_i \in R_{1,i} \cup B$, for $i = 1, \dots, |\mathbf{S}_1|$. Applying Claim D.1 and the fact $m(t_1^-) = m(0)$, which implies that $\sigma_M(m(t_1^-)) = \tau_M(m(t_1^-))$, we obtain

$$\begin{aligned}\sigma(R_{1,1}) &= \sigma\big(R\big(m(t_1^-), w_1, \varnothing, |\mathbf{S}_1|\big)\big)\\ &= R\big(\sigma_M\big(m(t_1^-)\big), w_1, \varnothing, |\mathbf{S}_1|\big)\\ &= R\big(\tau_M\big(m(t_1^-)\big), w_1, \varnothing, |\mathbf{S}_1|\big)\\ &= \tau\big(R\big(m(t_1^-), w_1, \varnothing, |\mathbf{S}_1|\big)\big)\\ &= \tau(R_{1,1}).\end{aligned} \tag{D.1}$$

Now recall that σ and τ preserve the order of $\overline{R} \setminus B$ and fix $\overline{R} \cap B$, so in particular they preserve the order of $R_{1,1} \setminus B \subset \overline{R} \setminus B$ and fix $R_{1,1} \cap B \subset \overline{R} \cap B$. Combining this with equation (D.1), we must have $\sigma(i) = \tau(i)$, for all $i \in R_{1,1}$. If $(\mathbf{S}_1)_1 \in R_{1,1}$, this implies that

$$\tau\big((\mathbf{S}_1)_1\big) = \sigma\big((\mathbf{S}_1)_1\big). \tag{D.2}$$

On the other hand, if $(\mathbf{S}_1)_1$ does not belong to $R_{1,1}$, then, from the definition of F, we must have $(\mathbf{S}_1)_1 \in F$. Since σ and τ fix the set F, we conclude that equation (D.2) must hold in all cases.

Proceeding inductively, and using the same argument, we obtain

$$\sigma(R_{1,i}) = \tau(R_{1,i}),$$

for $i = 1, \dots, |\mathbf{S}_1|$, and $\sigma(i) = \tau(i)$, for all $i \in \mathbf{S}_1^{\text{set}}$. It follows that $\sigma(\mathbf{S}_1) = \tau(\mathbf{S}_1)$. Combining this with the fact that $\sigma_M(m(t_1^-)) = \tau_M(m(t_1^-))$, and applying Claim D.1 twice, we obtain

$$\begin{aligned}\sigma(R_1') &= \sigma(R'(m(t_1^-), w_1, \mathbf{S}_1, \mathbf{q}_{\mathbf{S}_1}(t_1^-)))\\ &= R'(\sigma_M(m(t_1^-)), w_1, \sigma(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-))\\ &= R'(\tau_M(m(t_1^-)), w_1, \tau(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-))\\ &= \tau(R'(m(t_1^-), w_1, \mathbf{S}_1, \mathbf{q}_{\mathbf{S}_1}(t_1^-)))\\ &= \tau(R_1').\end{aligned} \tag{D.3}$$

Now recall that σ and τ preserve the order of $\overline{R} \setminus B$ and fix $\overline{R} \cap B$, so in particular they preserve the order of $R_1' \setminus B \subset \overline{R} \setminus B$ and fix $R_1' \cap B \subset \overline{R} \cap B$. Combining this with equation (D.3), we must have $\sigma(i) = \tau(i)$, for all $i \in R_1'$. Furthermore, recall that we also have that $\sigma(i) = \tau(i)$, for all $i \in \mathbf{S}_1^{\text{set}}$. If $D_1 \in R_1' \cup \mathbf{S}_1^{\text{set}}$, this implies that

$$\sigma(D_1) = \tau(D_1). \tag{D.4}$$

On the other hand, if D_1 does not belong to $R_1' \cup \mathbf{S}_1^{\text{set}}$, then, from the definition of F, we must have $D_1 \in F$. Since σ and τ fix the set F, we conclude that equation (D.4) must hold in all cases.

We now consider a memory update. Using the symmetry assumption, we have

$$\begin{aligned}\sigma_M(m(t_1)) &= \sigma_M(f_3(m(t_1^-), w_1, \mathbf{S}_1, \mathbf{q}_{\mathbf{S}_1}(t_1^-), D_1))\\ &= f_3(\sigma_M(m(t_1^-)), w_1, \sigma(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-), \sigma(D_1)).\end{aligned}$$

Then, since $\sigma_M(m(t_1^-)) = \tau_M(m(t_1^-))$, $\sigma(\mathbf{S}_1) = \tau(\mathbf{S}_1)$, and $\tau(D_1) = \sigma(D_1)$, we have

$$\begin{aligned}&f_3(\sigma_M(m(t_1^-)), w_1, \sigma(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-), \sigma(D_1))\\ &\quad = f_3(\tau_M(m(t_1^-)), w_1, \tau(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-), \tau(D_1)).\end{aligned}$$

Using the symmetry assumption once again, we obtain

$$\begin{aligned}&f_3(\tau_M(m(t_1^-)), w_1, \tau(\mathbf{S}_1), \mathbf{q}_{\mathbf{S}_1}(t_1^-), \tau(D_1))\\ &\quad = \tau_M(f_3(m(t_1^-), w_1, \mathbf{S}_1, \mathbf{q}_{\mathbf{S}_1}(t_1^-), D_1))\\ &\quad = \tau_M(m(t_1)).\end{aligned}$$

We conclude that

$$\sigma_M(m(t_1)) = \tau_M(m(t_1)).$$

Finally, since the memory states at time t_1 are still equal, we can proceed inductively by applying the same argument to obtain that, for $k = 2, \dots, c+1$, we have $\sigma(R_{k,i}) = \tau(R_{k,i})$ for $i = 1, \dots, |\mathbf{S}_k|$, and $\sigma(R_k') = \tau(R_k')$. It follows that $\sigma(\overline{R}) = \tau(\overline{R})$.

Acknowledgments. This work was supported in part by an MIT Jacobs Presidential Fellowship, NSF Grant CMMI-1234062 and ONR Grant N00014-17-1-2790.

REFERENCES

[1] ALON, N., LUBETZKY, E. and GUREL-GUREVICH, O. (2009). Choice-memory tradeoff in allocations. In 2009 50*th Annual IEEE Symposium on Foundations of Computer Science—FOCS* 2009 230–238. IEEE Computer Soc., Los Alamitos, CA. MR2648405 https://doi.org/10.1109/FOCS.2009.49

[2] AZAR, Y., BRODER, A. Z., KARLIN, A. R. and UPFAL, E. (1999). Balanced allocations. *SIAM J. Comput.* **29** 180–200. MR1710347 https://doi.org/10.1137/S0097539795288490

[3] BACCELLI, F. and BRÉMAUD, P. (2003). *Elements of Queueing Theory*: *Palm Martingale Calculus and Stochastic Recurrences*, 2nd ed. *Applications of Mathematics* (*New York*) **26**. Springer, Berlin. MR1957884 https://doi.org/10.1007/978-3-662-11657-9

[4] BADONNEL, R. and BURGESS, M. (2008). Dynamic pull-based load balancing for autonomic servers. In *Network Operations and Management Symposium* (*NOMS*).

[5] BRAMSON, M., LU, Y. and PRABHAKAR, B. (2013). Decay of tails at equilibrium for FIFO join the shortest queue networks. *Ann*. *Appl*. *Probab*. **23** 1841–1878. MR3114919 https://doi.org/10.1214/12-AAP888

[6] FEITELSON, D. and JETTE, M. A. (1997). Improved utilization and responsiveness with gang scheduling. In *IPPS* '97 *Proceedings of the Job Scheduling Strategies for Parallel Processing* 238–261.

[7] GAMARNIK, D., TSITSIKLIS, J. N. and ZUBELDIA, M. (2018). Delay, memory, and messaging tradeoffs in distributed service systems. *Stoch*. *Syst*. **8** 45–74. MR3775991 https://doi.org/10.1287/stsy.2017.0008

[8] HARCHOL-BALTER, M., CROVELLA, M. E. and MURTA, C. D. (1999). On choosing a task assignment policy for a distributed server system. *J*. *Parallel Distrib*. *Comput*. **59** 204–228.

[9] HELLEMANS, T. and VAN HOUDT, B. (2018). On the power-of-d-choices with least loaded server selection. *Proc*. *ACM Meas*. *Anal*. *Comput*. *Syst*. **2** Article No. 27.

[10] LENZEN, C. and WATTENHOFER, R. (2016). Tight bounds for parallel randomized load balancing. *Distrib*. *Comput*. **29** 127–142. MR3488178 https://doi.org/10.1007/s00446-014-0225-4

[11] 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. *Perform*. *Eval*. **68** 1056–1071.

[12] MITZENMACHER, M. D. (1996). *The Power of Two Choices in Randomized Load Balancing*. ProQuest LLC, Ann Arbor, MI. Thesis (Ph.D.)—Univ. California, Berkeley. MR2695522

[13] MITZENMACHER, M., PRABHAKAR, B. and SHAH, D. (2002). Load balancing with memory. In *Proceedings of the* 43*rd Annual IEEE Symposium on Foundations of Computer Science*.

[14] MUKHERJEE, D., BORST, S., VAN LEEUWAARDEN, J. and WHITING, P. (2016). Universality of power-of-d load balancing schemes. In *Workshop on Mathematical Performance Modeling and Analysis* (*MAMA*).

[15] STAMOULIS, G. D. and TSITSIKLIS, J. N. (1991). Optimal distributed policies for choosing among multiple servers. In *Proceedings of the* 30*th Conference on Decision and Control* 815–820.

[16] STOLYAR, A. L. (2015). Pull-based load distribution in large-scale heterogeneous service systems. *Queueing Syst*. **80** 341–361. MR3367704 https://doi.org/10.1007/s11134-015-9448-8

[17] VVEDENSKAYA, N. D., DOBRUSHIN, R. L. and KARPELEVICH, F. I. (1996). Queueing system with selection of the shortest of two queues: An asymptotic approach. *Probl*. *Inf*. *Transm*. **32** 15–27. MR1384927

[18] WINSTON, W. (1977). Optimality of the shortest line discipline. *J*. *Appl*. *Probab*. **14** 181–189. MR0428516 https://doi.org/10.1017/s0021900200104772

[19] YING, L., SRIKANT, R. and KANG, X. (2015). The power of slightly more than one sample in randomized load balancing. In *Proceedings of the IEEE Conference on Computer Communications*.