Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times

ISAAC GROSOF, Carnegie Mellon University, USA
ZIV SCULLY, Carnegie Mellon University, USA
MOR HARCHOL-BALTER, Carnegie Mellon University, USA

Load balancing systems, comprising a central dispatcher and a scheduling policy at each server, are widely used in practice, and their response time has been extensively studied in the theoretical literature. While much is known about the scenario where the scheduling at the servers is First-Come-First-Served (FCFS), to minimize mean response time we must use Shortest-Remaining-Processing-Time (SRPT) scheduling at the servers. Much less is known about dispatching polices when SRPT scheduling is used. Unfortunately, traditional dispatching policies that are used in practice in systems with FCFS servers often have poor performance in systems with SRPT servers. In this paper, we devise a simple fix that can be applied to any dispatching policy. This fix, called *guardrails*, ensures that the dispatching policy yields optimal mean response time under heavy traffic when used in a system with SRPT servers. *Any* dispatching policy, when augmented with guardrails, becomes heavy-traffic optimal. Our results yield the first analytical bounds on mean response time for load balancing systems with SRPT scheduling at the servers.

CCS Concepts: • **General and reference → Performance**; • **Mathematics of computing → Queueing theory**; • **Networks → Network performance modeling**; • **Theory of computation →** *Routing and network design problems*; • **Computing methodologies →** *Model development and analysis*; • **Software and its engineering →** *Scheduling*.

Additional Key Words and Phrases: load balancing; dispatching; scheduling; SRPT; response time; latency; sojourn time; heavy traffic

ACM Reference Format:
Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 2, Article 42 (June 2019), 31 pages. https://doi.org/10.1145/3326157

1 INTRODUCTION

Load balancers are ubiquitous throughout computer systems. They act as a front-end to web server farms, distributing HTTP requests to different servers. They likewise act as a front-end to data centers and cloud computing pools, where they distribute requests among servers and virtual machines.

In this paper, we consider the immediate dispatch load balancing model, where each arriving job is immediately dispatched to a server, as shown in Figure 1. The system has two decision points:

(1) A *dispatching policy* decides how to distribute jobs across the servers.
(2) A *scheduling policy* at each server decides which job to serve among those at that server.

Authors' address: Carnegie Mellon University, Computer Science Department, 5000 Forbes Ave, Pittsburgh, PA 15213, USA, {igrosof, zscully, harchol}@cs.cmu.edu.

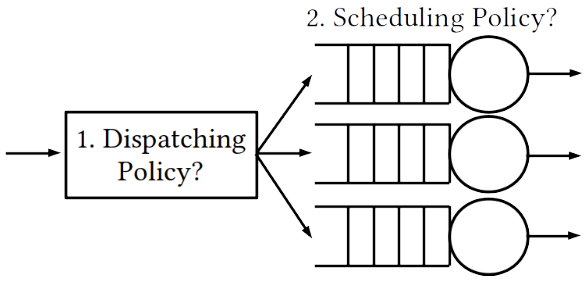


Fig. 1. Two decision points within a load balancing system: (1) Pick the dispatching policy. (2) Pick the scheduling policy for the servers.

We ask:

> What (1) dispatching policy and (2) scheduling policy should we use to *minimize mean response time* of jobs?

We assume that the job arrival process is Poisson and that job sizes are i.i.d. from a general size distribution. We assume jobs are preemptible with no loss of work. Finally, we assume that job sizes are known at the time the job arrives in the system.

With these assumptions, the scheduling question turns out to be easy to answer: use Shortest-Remaining-Processing-Time (SRPT) at the servers. No matter what dispatching decisions are made, if we consider the sequence of jobs dispatched to a particular server, the policy which minimizes mean response time for that server must be to schedule those jobs in SRPT order. This follows from from the optimality of SRPT for arbitrary arrival sequences [25]. SRPT scheduling is in fact already used in backend servers [16, 23]. Thus, in the remainder of this paper we assume SRPT is used at the servers.

The question remains: What dispatching policy minimizes mean response time given SRPT service at the servers? While many dispatching policies have been considered in the literature, they have mostly been considered in the context of First-Come-First-Served (FCFS) or Processor-Sharing (PS) scheduling at the servers. Popular dispatching policies include Random routing [14, 22], Least-Work-Left (LWL) [7, 14, 15], Join-Shortest-Queue (JSQ) [6, 12, 26, 29], JSQ-d [7, 19, 22, 24], Size-Interval-Task-Assignment (SITA) [4, 10, 14], Round-Robin (RR) [14, 21], and many more [2, 5, 8, 31]. However, only the simplest of these policies, such as Random and RR, have been studied for SRPT servers [9, 13].

One might hope that the same policies that yield low mean response time when servers use FCFS scheduling would also perform well when servers use SRPT scheduling. Unfortunately, this does not always hold. For example, when the servers use FCFS, it is well-known that LWL dispatching, which sends each job to the server with the least remaining work, outperforms Random dispatching, which sends each job to a randomly chosen server. (We write this as LWL/FCFS outperforms Random/FCFS.) However, the opposite can happen when the servers use SRPT: as shown in the scenario in Figure 2, Random/SRPT can outperform LWL/SRPT. Moreover, the performance difference is highly significant: Random/SRPT outperforms LWL/SRPT by a factor of 5 or more under heavy load. This means that LWL is making serious mistakes in dispatching decisions. We can therefore see that the heuristics that served us well for FCFS servers can steer us awry when we use SRPT servers.

In this paper, we introduce *guardrails*, a new technique for creating dispatching policies. Given an arbitrary dispatching policy P, applying guardrails results in an improved policy Guarded-P

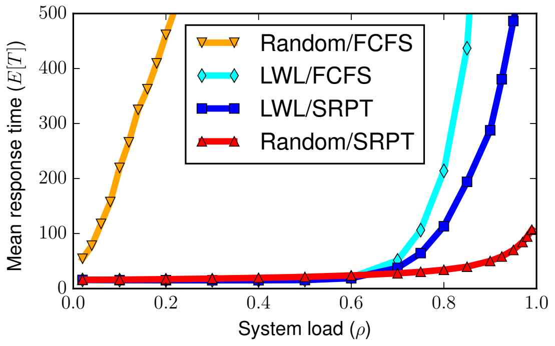


Fig. 2. Two dispatching policies: Random and LWL. Two scheduling policies: FCFS and SRPT. FCFS scheduling at the servers yields higher mean response time as a function of load, compared with SRPT scheduling at the servers. Random dispatching is worse than LWL dispatching under FCFS scheduling at the servers, but Random dispatching is better than LWL dispatching under SRPT scheduling at the servers. Simulation uses $k = 10$ servers. Size distribution shown is Bimodal with jobs of size 1 with probability 99.95% and jobs of size 1000 with probability 0.05%.

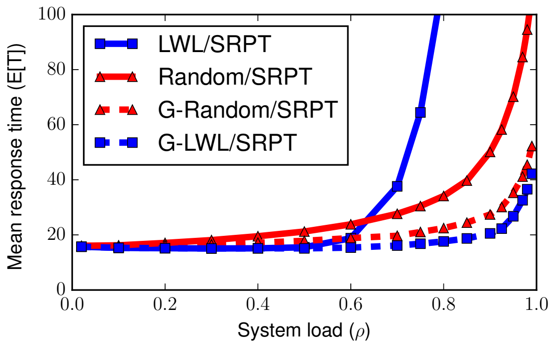


Fig. 3. Adding guardrails to LWL yields much lower mean response time as a function of load. Guardrails yield a factor of 3 improvement even at $\rho = 0.8$, and a factor of 7 improvement at $\rho = 0.9$. Adding guardrails to Random also yields significantly lower mean response time as a function of load. Simulation uses $k = 10$ servers. Size distribution shown is Bimodal with jobs of size 1 with probability 99.95% and jobs of size 1000 with probability 0.05%. The guardrails have tightness $g = 2$.

(G-P). We prove that the improved policy G-P has asymptotically optimal mean response time in the heavy traffic limit, no matter what the initial policy P is. We also show empirically that adding guardrails to a policy almost always decreases its mean response time (and never significantly increases it), even outside the heavy-traffic regime.

As an example of the power of guardrails, Figure 3 shows the performance of guarded versions of LWL and Random, namely G-LWL and G-Random. The guardrails stop LWL from making serious mistakes and dramatically improve its performance. Random dispatching also benefits from guardrails. Moreover, the guarded policies have a theoretical guarantee: In the limit as load $\rho \longrightarrow 1$, G-Random/SRPT and G-LWL/SRPT converge to the optimal mean response time. In contrast, unguarded Random/SRPT is a factor of k worse than optimal in the $\rho \longrightarrow 1$ limit, where k is the number of servers.

This paper makes the following contributions:

- In Section 2, we introduce guardrails, a technique for improving any dispatching policy.
- In Section 4, we bound the mean response time of any guarded dispatching policy when paired with SRPT scheduling at the servers. Using that bound, we prove that any guarded policy has asymptotically optimal mean response time as load $\rho \to 1$, subject to a technical condition on the job size distribution roughly equivalent to finite variance.
- In Section 5, we consider a wide variety of common dispatching policies. We empirically show that guardrails improve most of these at all loads.
- In Section 6, we discuss practical considerations and extensions of guardrails, including
  - guardrails for large systems, which may have multiple dispatchers and network delays;
  - guardrails for scheduling policies other than SRPT; and
  - guardrails for heterogeneous servers.

We give a more technical summary of our theoretical results and review related work in Section 3.

2 LOAD BALANCING GUARDRAILS

2.1 What are Guardrails?

Traditional dispatching policies aim to equalize load at each server. However, minimizing mean response time requires more than balancing load: we also need to find a way to favor small jobs. Given that every server uses SRPT scheduling, if we can spread out the small jobs across the servers, then we ensure that the maximum possible number of servers are working on the smallest jobs available. Our idea is to take any dispatching policy and add "guardrails" that force it to spread out small jobs across the servers.

In the discussion above, "small" is a relative term. Whatever the size of the smallest jobs currently in the system, we would like to spread out jobs near that size across the servers. To do this, we define the *rank* of a job of size x to be

$$r = \lfloor \log_c x \rfloor, \tag{1}$$

where $c > 1$ is a constant called the *guardrail rank width* (see Section 2.1.1). The idea of guardrails is to spread out the jobs within a rank r across the servers, doing so separately for each rank r. To do so, for each rank r and each server s, the dispatcher stores a *guardrail work counter* G_s^r. When dispatching a job of size x to server s, the dispatcher increases G_s^r by x, with r given by (1).[1] Guardrails are a set of constraints which ensure that no two rank r work counters are ever too far apart.

Definition 1. A dispatching policy *satisfies guardrails with tightness g* if at all times

$$|G_s^r - G_{s'}^r| \le gc^{r+1}$$

for all ranks r and all pairs of servers s and s', where $c > 1$ is the same constant as in (1). The tightness can be any constant $g \ge 1$.

We sometimes say that a particular dispatching decision satisfies (respectively, violates) guardrails if it satisfies (respectively, violates) the constraints imposed by Definition 1.

2.1.1 *Choosing the Guardrail Rank Width c.*

The choice of c in (1) heavily affects the performance of policies satisfying guardrails.

- If c is too large, then guardrails may not differentiate between jobs of different sizes.
- If c is too small, then guardrails may misguidedly differentiate between jobs of similar sizes. This could allow one server to receive multiple small jobs of different ranks while another receives none.

[1]The dispatcher also occasionally decreases work counters, as explained in Section 2.3.

To balance this tradeoff, we set c to be a function of load ρ:

$$c = 1 + \frac{1}{1 + \ln \frac{1}{1-\rho}}. \tag{2}$$

This particular value of c is chosen to enable the heavy-traffic optimality proof for any dispatching policy satisfying guardrails.

2.2 Guarded Policies: How to Augment Dispatching Policies with Guardrails

Guardrails as described in Definition 1 are a set of constraints on dispatching policies that we will use to guarantee bounds on mean response time (see Section 4). However, the constraints alone do not give a complete dispatching policy. To define a concrete dispatching policy satisfying guardrails, we start with an arbitrary dispatching policy P and augment it to create a new policy, called *Guarded-P* (G-P), which satisfies guardrails.

Roughly speaking, G-P tries to dispatch according to P, but if dispatching to P's favorite server would violate guardrails, G-P considers P's second-favorite server, and so on. Below are guarded versions of some common dispatching policies:

- G-Random dispatches to a random server among those which satisfy guardrails.
- G-LWL dispatches to the server with the least remaining work among those which satisfy guardrails.
- Round-Robin (RR) can be seen as always dispatching to the server that has least recently received a job, so G-RR dispatches to the server that has least recently received a job among those which satisfy guardrails.

Given an arbitrary dispatching policy P, Algorithm 1 formally defines G-P. We assume that P is specified by procedure $\textsc{Dispatch}^{\mathrm{P}}$ which, when passed a job of size x and a set of servers $\mathcal{S}$, returns a server in $\mathcal{S}$ to which P would dispatch a job of size x. The key to Algorithm 1 is that instead of calling $\textsc{Dispatch}^{\mathrm{P}}$ with the set of all servers, we pass it a restricted set of servers $\mathcal{S}_{\text{safe}} \subseteq \mathcal{S}$ such that dispatching to any server in $\mathcal{S}_{\text{safe}}$ will satisfy guardrails. $\mathcal{S}_{\text{safe}}$ is never empty because $x \leq gc^{r+1}$, so $\mathcal{S}_{\text{safe}}$ will always contain the server s' of minimal $G^r_{s'}$.

Algorithm 1 is phrased in terms of for loops over all ranks r. While there are infinitely many ranks in theory, it is simple to represent all of the work counters in finite space by representing most of them implicitly.

2.3 Resets

Algorithm 1 includes a procedure, $\textsc{Reset}^{\text{G-P}}$, which we have not yet explained. As defined so far, guardrails effectively spread out small jobs across the servers, but they have an unfortunate side effect: they sometimes prevent dispatches to empty servers. This is because the work counters G^r_s as defined so far depend only on the dispatching history, not the current server state.

Because dispatching to empty servers is desirable, we would like to ensure that dispatching to an empty server never violates guardrails. We accomplish this by having servers *reset* whenever they becomes empty. When a server s resets, for each rank r, we decrease G^r_s to match the minimum among all rank r work counters. Because all rank r jobs have size less than gc^{r+1}, by Definition 1, dispatching to a server that has just reset will never violate guardrails.

3 TECHNICAL SUMMARY

3.1 System Model

We will study a k-server load balancing system with Poisson arrivals at rate λ jobs per second and job size distribution X. Our optimality results (Theorem 2) assume that either

Algorithm 1 Guarded-P (G-P)

```
*Given:* dispatching policy P, tightness $g \geq 1$, set of servers $\mathcal{S}$, and rank width $c = 1 + \frac{1}{1+\ln\frac{1}{1-\rho}}$
*Call when:* the system starts
  **procedure** INITIALIZE$^{\text{G-P}}$()
      **for** each rank $r$ and each server $s \in \mathcal{S}$ **do**
          $G_s^r \leftarrow 0$
*Call when:* a job of size $x$ arrives
  **procedure** DISPATCH$^{\text{G-P}}(x)$
      $r \leftarrow \lfloor \log_c x \rfloor$
      $G_{\min} \leftarrow \min_{s' \in \mathcal{S}} G_{s'}^r$
      $\mathcal{S}_{\text{safe}} \leftarrow \{s' \in \mathcal{S} \mid G_{s'}^r + x \leq G_{\min} + gc^{r+1}\}$
      $s \leftarrow$ DISPATCH$^{\text{P}}(x, \mathcal{S}_{\text{safe}})$
      $G_s^r \leftarrow G_s^r + x$
      **return** $s$
*Call when:* server $s$ becomes empty
  **procedure** RESET$^{\text{G-P}}(s)$
      **for** each rank $r$ **do**
          $G_s^r \leftarrow \min_{s' \in \mathcal{S}} G_{s'}^r$
```

- X has bounded maximum size or
- the tail of X has upper Matuszewska index[2] less than -2.

This disjunctive assumption is roughly equivalent to finite variance. We adopt the convention that each of the k servers serves jobs at speed $1/k$. As a result, a job of size x requires kx service time to complete. We have chosen to define the speed of a server this way because we will later compare the k-server system with a single server system of speed 1, and this convention allows us to directly apply standard results on single server systems. We define the system load ρ for both a single-server system and the k-server system by

$$\rho = \lambda \mathbf{E}[X] < 1.$$

Load does not depend on k because the total service rate of all k servers combined is 1.

Throughout, we assume that the dispatching policy is a guarded policy, as defined in Algorithm 1. We consider two different scheduling policies that might be used at the servers:

SRPT The policy that serves the job of least remaining size.

Priority-c The preemptive class-based priority policy in which a job's class is its *rank*, as defined by (1). That is, a job of size x has rank $r = \lfloor \log_c x \rfloor$, and Priority-$c$ serves the job of minimal rank. Within each rank, jobs are served FCFS.

3.2 Theorem Overview

Our overall goal is to prove, for any dispatching policy P, the asymptotic optimality of the policy Guarded-P (G-P) with respect to mean response time, given SRPT scheduling at the servers. We refer to this joint dispatch/scheduling policy as G-P/SRPT.

[2]See Appendix A.

Rather than studying G-P/SRPT directly, we instead bound mean response time under G-P/Priority-c. By the optimality of SRPT scheduling [25], the mean response time under G-P/Priority-c gives an upper bound on the mean response time under G-P/SRPT.

Theorem 1. For any dispatching policy P, consider the policy G-P with tightness g. The expected response time for a job of size x under G-P/Priority-c is bounded by

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^r})(1-\rho_{c^{r+1}})} + \frac{(4c+2)gk\frac{c^{r+1}}{c-1} + kx}{(1-\rho_{c^r})},$$

where

- $f_X(\cdot)$ is the probability density function of X,
- c is the guardrail rank width
- $r = \lfloor \log_c x \rfloor$ is the rank of a job of size x, and
- $\rho_y = \lambda \int_0^y t f_X(t)\, dt$ is the load due to jobs of size $\leq y$.

We prove Theorem 1 in Section 4.3.

Using the bound in Theorem 1, we show that the mean response time of the G-P/Priority-c system converges to that of a single-server SRPT system in the heavy-traffic limit.

Theorem 2. Consider a single-server SRPT system whose single server is k times as fast as each server in the load balancing system. For any dispatching policy P, consider the policy G-P with any constant tightness. Then for any size distribution X which is either (i) bounded or (ii) unbounded with tail having upper Matuszewska index[3] less than -2, the mean response times of G-P/SRPT, G-P/Priority-c, and (single-server) SRPT converge as load approaches capacity:

$$\lim_{\rho\to 1} \frac{\mathbf{E}[T]^{\text{G-P/SRPT}}}{\mathbf{E}[T]^{\text{SRPT}}} = \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} = 1.$$

We prove Theorem 2 in Section 4.4.

Theorem 2 relates the mean response times of G-P/SRPT and single-server SRPT, which has the optimal mean response time among all single-server policies [25]. But a single-server system can simulate a load balancing system running any joint dispatching/scheduling policy P′/S′. As a result, the mean response time under single-server SRPT is a lower bound for the mean response time under P′/S′.

Using that bound, Theorem 2 implies the following relationship between the mean response times of G-P/SRPT and P′/S′.

Corollary 1. For any dispatching policy P consider the policy G-P with any constant tightness. Consider any joint dispatching/scheduling policy P′/S′. Then for any size distribution X which is either (i) bounded or (ii) unbounded with tail having upper Matuszewska index[3] less than -2, the mean response times of G-P/SRPT and G-P/Priority-c are at least as small as the mean response time of P′/S′ as load approaches capacity:

$$\lim_{\rho\to 1} \frac{\mathbf{E}[T]^{\text{G-P/SRPT}}}{\mathbf{E}[T]^{\text{P}'/\text{S}'}} = \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{P}'/\text{S}'}} \leq 1.$$

[3]See Appendix A.

3.3 Relationship to Prior Work

Our guardrails provide the first mechanism to augment an arbitrary dispatching policy to ensure size balance at all job size scales. Moreover, we give the first bound on the mean response time of load balancing systems with SRPT scheduling at the servers. Using this bound, we prove that guarded dispatching policies have asymptotically optimal mean response time in the $\rho \to 1$ limit for any constant number of servers. Our guarded policies are the first dispatching policies known to have this property for general job size distributions.

We are not the first to consider load balancing systems with SRPT scheduling at the servers. Avrahami and Azar [3] consider a problem analogous to ours but in a *worst-case* setting, assuming adversarial arrival times and job sizes, in contrast to our stochastic setting. Their dispatching policy, which they call IMD, divides jobs into size ranks in a manner similar to our size ranks, except with the width of each rank set to $c = 2$. IMD dispatches each rank r job to the server that has received the least work of rank r jobs in the past. Put another way, IMD is the policy that keeps maximally tight guardrails with no other underlying policy. Avrahami and Azar prove that IMD is $O(\log P)$ competitive with an optimal migratory offline algorithm, where P is the ratio between the largest and smallest job sizes in this system. Note that P can be arbitrarily large for general job size distributions. Unfortunately, the $O(\log P)$ competitive ratio is optimal for any online dispatching policy in the worst-case setting [18]. Our result is much stronger thanks to our stochastic setting.

Down and Wu [9] also consider a stochastic setting with SRPT scheduling at the servers, and they also propose a dispatching policy that balances jobs of different sizes across the servers. Their analysis does not result in any formula for mean response time but instead uses a diffusion limit argument to show optimality in heavy traffic. However, this limits their results to discrete job size distributions, thus exlcuding many practically important continuous job size distributions. In fact, Down and Wu [9, Section 5] observe empirically that their policy performs poorly for Bounded Pareto job size distributions. In contrast, our analysis shows that any guarded dispatching policy is heavy-traffic optimal for general job size distributions, including Bounded Pareto (see Figure 4). Finally, the Down and Wu [9] result provides no insight into mean response time outside of the heavy traffic regime, whereas we derive a mean response time bound that is valid for all loads.

4 ANALYSIS OF GUARDED POLICIES

In this section, we analytically bound the mean response time of a load balancing system using an arbitrary guarded dispatching policy G-P paired with SRPT scheduling. We then show that our bound implies that G-P/SRPT minimizes mean response time in heavy traffic.

4.1 Preliminaries and Notation

We use a tagged job analysis: we analyze the expected response time of a particular "tagged" job, which we call j, arriving to a steady-state system. The expected response time of j is equal to the system's mean response time by the PASTA property [30].

Instead of studying G-P/SRPT directly, we analyze G-P/Priority-c, which yields an upper bound on the mean response time under G-P/SRPT. Studying Priority-c simplifies the analysis because the priority classes of Priority-c match the ranks used by guardrails.

Suppose that j has rank r and is dispatched to server s. Under Priority-c scheduling, there are two types of work that might delay job j:

- The *current relevant work* at server s when j arrives. This is the total amount of remaining work at server s due to jobs of rank $\le r$.
- The *future relevant work* due to arriving jobs dispatched to server s while j is in the system. These are the jobs dispatched to s of rank $< r$ (that is, rank $\le r - 1$).

We use the following notation, where "rank r work" denotes work due to jobs of rank r.

- $W_s^r(t)$ denotes the current amount of rank r work at server s at time t.
- $V_s^r(t)$ denotes the total amount of rank r work that has ever been dispatched to server s up to time t. In particular, the amount of rank r work dispatched to s during a time interval (t_1, t_2) is $V_s^r(t_2) - V_s^r(t_1)$.
- $G_s^r(t)$ denotes the rank r guardrail work counter for server s at time t (see Algorithm 1). Specifically, $G_s^r(t)$ is defined as follows:[4]
  - If a rank r job of size x is dispatched to server s at time t, we set $G_s^r(t^+) = G_s^r(t^-) + x$.
  - If a server s becomes empty of all jobs at time t, we set $G_s^r(t^+) = \min_{s'} G_{s'}^r(t^-)$, where s' ranges over all servers. We call this a *reset* of server s.
  - Otherwise, $G_s^r(t)$ does not change.

We write $W_s^{\le r}(t)$, $V_s^{\le r}(t)$, and $G_s^{\le r}(t)$ to denote the corresponding quantities where we consider all ranks $\le r$, rather than just rank r, and similarly for superscript $< r$.

Occasionally, we will be talking about the *total* work in the system, or the *total* work that has arrived, summed over all servers. In that case, we will drop the subscript s, writing $W^{\le r}(t)$ or $V^{\le r}(t)$. Finally, we write $W^{\le r}$ to denote the stationary distribution of the amount of rank $\le r$ work in the whole system.

4.2 Bounding Response Time: Key Steps

Our goal in this section is to bound the expected response time of a tagged job j under G-P/Priority-c. We assume that j has size x and rank $r = \lfloor \log_c x \rfloor$. We first bound current relevant work, then move on to bound future relevant work.

We begin by showing that guardrails ensure that any two servers have a similar amount of remaining rank $\le r$ work.

Lemma 1. For any dispatching policy P, consider the dispatching policy G-P with tightness g. In a G-P/Priority-c system, the difference in remaining rank $\le r$ work between any two servers s and s' at any time t is bounded by

$$W_s^{\le r}(t) - W_{s'}^{\le r}(t) \le \frac{2gc^{r+2}}{c-1},$$

where c is the guardrail rank width.

We prove Lemma 1 in Section 4.3.1.

Roughly speaking, Lemma 1 shows that guarded policies do a good job of spreading out rank $\le r$ work across the servers. This is important because if the rank $\le r$ work is spread out well, then whenever there is a large amount of rank $\le r$ work in the system, *all the servers* are doing rank $\le r$ work. This allows us to bound the amount of rank $\le r$ work in the k-server G-P/Priority-c system in terms of the remaining rank $\le r$ work in an M/G/1/Priority-c system with a single server that runs k times as fast.

Lemma 2. For any dispatching policy P, consider the dispatching policy G-P with tightness g. The total amount of remaining rank $\le r$ work in a G-P/Priority-c system is stochastically bounded relative to the remaining rank $\le r$ work in a M/G/1/Priority-c system whose server runs k times as fast:

$$W^{\le r} \le_{\text{st}} W^{\le r}_{\text{M/G/1/Priority-}c} + \frac{2gkc^{r+2}}{c-1},$$

where c is the guardrail rank width.

[4]The notations t^- and t^+ below refer to "just before" and "just after" time t. More formally, they refer to the left and right limits, respectively, of an expression that is piecewise-continuous in t.

We prove Lemma 2 in Section 4.3.2.

Combining Lemmas 1 and 2 yields a bound on the amount of remaining rank $\le r$ work at any server s, thus bounding the current relevant work.

We now turn to bounding future relevant work. Suppose that the tagged job j is dispatched to server s. The fact that guardrails spread out relevant work across the servers means that while j is in the system s will not receive much more rank $< r$ work than other servers, thus bounding future relevant work. Combining this with our bound on current relevant work yields the following bound on j's response time.

Lemma 3. For any dispatching policy P, consider the dispatching policy G-P with tightness g. In a G-P/Priority-c system, the response time of a job of size x is stochastically bounded by

$$T(x) \le_{\text{st}} B_{<r}\left(W^{\le r}_{\text{M/G/1/Priority-}c} + \frac{(4c+2)gkc^{r+1}}{c-1} + kx\right),$$

where c is the guardrail rank width, $r = \lfloor \log_c x \rfloor$ is the rank of the job, and $B_{<r}(w)$ is the length of a busy period comprising only jobs of rank $< r$ started by work w.

We prove Lemma 3 in Section 4.3.3.

Taking expectations in Lemma 3 and applying the well-known formula for $\mathbf{E}[W^{\le r}_{\text{M/G/1/Priority-}c}]$, we obtain Theorem 1.

Theorem 1. *For any dispatching policy P, consider the policy G-P with tightness g. The expected response time for a job of size x under G-P/Priority-c is bounded by*

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \le \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^r})(1-\rho_{c^{r+1}})} + \frac{(4c+2)gk\frac{c^{r+1}}{c-1} + kx}{(1-\rho_{c^r})},$$

where

- *$f_X(\cdot)$ is the probability density function of X,*
- *c is the guardrail rank width*
- *$r = \lfloor \log_c x \rfloor$ is the rank of a job of size x, and*
- *$\rho_y = \lambda \int_0^y t f_X(t)\,dt$ is the load due to jobs of size $\le y$.*

We prove Theorem 1 in Section 4.3.4.

4.3 Bounding Response Time: Proofs

4.3.1 Proof of Lemma 1.

Lemma 1. *For any dispatching policy P, consider the dispatching policy G-P with tightness g. In a G-P/Priority-c system, the difference in remaining rank $\le r$ work between any two servers s and s' at any time t is bounded by*

$$W_s^{\le r}(t) - W_{s'}^{\le r}(t) \le \frac{2gc^{r+2}}{c-1},$$

where c is the guardrail rank width.

Proof. Let t_0 be the most recent time up to time t when server s was empty of rank $\le r$ work. Note that t_0 may equal t. We will bound the difference in rank $\le r$ work at the two servers at time t by comparison with time t_0.

The remaining rank $\le r$ work present at time t is

(i) the remaining rank $\le r$ work present at time t_0
(ii) plus rank $\le r$ work due to arrivals in the interval $[t_0, t]$

(iii) minus rank $\leq r$ work processed during the interval.

We consider these three quantities first for server s, then for server s'.

We begin with server s:

(i) By the definition of t_0, there is no remaining rank $\leq r$ work at server s.
(ii) The amount of work that arrives to server s over the interval $[t_0, t]$ is $V_s^{\leq r}(t) - V_s^{\leq r}(t_0)$.
(iii) The amount of rank $\leq r$ work processed during the interval $[t_0, t]$ is equal to $\frac{t-t_0}{k}$, because the server s processes work at speed $1/k$, s has rank $\leq r$ work available throughout the interval, and the Priority-c scheduling policy always prioritizes lower rank work.

These quantities give us the remaining rank $\leq r$ work at server s at time t:

$$W_s^{\leq r}(t) = (V_s^{\leq r}(t) - V_s^{\leq r}(t_0)) - \frac{t - t_0}{k}. \tag{3}$$

Because server s was never empty at any time during $(t_0, t]$, the guardrail work counters were never reset to the system-wide minimums during $[t_0, t]$. As a result, the changes in $G_s^{\leq r}(\cdot)$ and $V_s^{\leq r}(\cdot)$ over the interval $[t_0, t]$ must be equal. We can apply this fact to (3):

$$W_s^{\leq r}(t) = (G_s^{\leq r}(t) - G_s^{\leq r}(t_0)) - \frac{t - t_0}{k}. \tag{4}$$

We now turn to server s':

(i) The remaining rank $\leq r$ work present at server s' at time t_0 is non-negative.
(ii) The amount of work that arrives to server s over the interval $[t_0, t]$ is $V_{s'}^{\leq r}(t) - V_{s'}^{\leq r}(t_0)$.
(iii) The amount of rank $\leq r$ work processed over the interval $[t_0, t]$ is at most $\frac{t-t_0}{k}$.

Therefore, we may lower bound the remaining rank $\leq r$ work at server s' at time t:

$$W_{s'}^{\leq r}(t) \geq (V_{s'}^{\leq r}(t) - V_{s'}^{\leq r}(t_0)) - \frac{t - t_0}{k}. \tag{5}$$

The change in $G_{s'}^{\leq r}(\cdot)$ over the interval $[t_0, t]$ is no more than the change in $V_{s'}^{\leq r}(\cdot)$ over the same interval, since any reset to the system-wide minimum can only lead to a decrease in $G_{s'}^{\leq r}(t)$. We can apply this fact to (5):

$$W_{s'}^{\leq r}(t) \geq (G_{s'}^{\leq r}(t) - G_{s'}^{\leq r}(t_0)) - \frac{t - t_0}{k}. \tag{6}$$

Combining (4), (6), and the guardrail constraint in Definition 1 yields the desired bound:

$$\begin{aligned}
W_s^{\leq r}(t) - W_{s'}^{\leq r}(t) &\leq (G_s^{\leq r}(t) - G_s^{\leq r}(t_0)) - (G_{s'}^{\leq r}(t) - G_{s'}^{\leq r}(t_0)) \\
&\leq |G_s^{\leq r}(t) - G_{s'}^{\leq r}(t)| + |G_s^{\leq r}(t_0) - G_{s'}^{\leq r}(t_0)| \\
&= \sum_{q=-\infty}^{r} (|G_s^q(t) - G_{s'}^q(t)| + |G_{s'}^q(t_0) - G_s^q(t_0)|) \\
&\leq \sum_{q=-\infty}^{r} 2gc^{q+1} \\
&= \frac{2gc^{r+2}}{c-1}.
\end{aligned}$$

□

4.3.2 *Proof of Lemma 2.*

LEMMA 2. *For any dispatching policy P, consider the dispatching policy G-P with tightness g. The total amount of remaining rank $\leq r$ work in a G-P/Priority-c system is stochastically bounded relative* *to the remaining rank $\leq r$ work in a M/G/1/Priority-c system whose server runs k times as fast:*

$$W^{\leq r} \leq_{\text{st}} W^{\leq r}_{\text{M/G/1/Priority-}c} + \frac{2gkc^{r+2}}{c-1},$$

where c is the guardrail rank width.

Proof. We consider two coupled systems receiving the same arrivals:

- a G-P/Priority-c system where each of the k servers runs at speed $1/k$, and
- a M/G/1/Priority-c system where the single server runs at speed 1.

We will refer to the total amount of remaining rank $\leq r$ work in the G-P/Priority-c system as $W^{\leq r}(t)$, and the total amount of remaining rank $\leq r$ work in the M/G/1/Priority-c system as $W^{\leq r}_{\text{M/G/1/Priority-}c}(t)$.

It suffices to show that at any time t, we have the following bound on the difference in the total amounts of remaining rank $\leq r$ work between the two systems:

$$W^{\leq r}(t) \leq W^{\leq r}_{\text{M/G/1/Priority-}c}(t) + \frac{2gkc^{r+2}}{c-1}. \tag{7}$$

To prove (7), we consider two cases:

(i) At least one server in the G-P/Priority-c system that has no remaining rank $\leq r$ work at time t.
(ii) All servers in the G-P/Priority-c system have remaining rank $\leq r$ work at time t.

In case (i), suppose server s' in the G-P/Priority-c system has no remaining rank $\leq r$ work at time t. By Lemma 1, we know that at all servers s,

$$W^{\leq r}_s(t) = W^{\leq r}_s(t) - W^{\leq r}_{s'}(t) \leq \frac{2gc^{r+2}}{c-1}.$$

Summing over all k servers implies (7).

We now turn to case (ii). Let t_0 be the most recent time before t when a G-P/Priority-c server had no remaining rank $\leq r$ work. Note that case (i) applies at time t_0. Therefore, it suffices to show that the difference in remaining rank $\leq r$ work between the two systems is no more at time t than at time t_0:

$$W^{\leq r}(t) - W^{\leq r}_{\text{M/G/1/Priority-}c}(t) \leq W^{\leq r}(t_0) - W^{\leq r}_{\text{M/G/1/Priority-}c}(t_0). \tag{8}$$

By definition of t_0, for the duration of entire time interval (t_0, t), each of the k servers in the G-P/Priority-c system processes rank $\leq r$ work at speed $1/k$, for a total of $t - t_0$ work. This is at least as much rank $\leq r$ work as the M/G/1/Priority-c system processes during (t_0, t), because the single server's speed is 1. Due to coupling, the two systems receive the same amount of rank $\leq r$ work during (t_0, t), implying (8). □

4.3.3 Proof of Lemma 3.

Lemma 3. *For any dispatching policy P, consider the dispatching policy G-P with tightness g. In a G-P/Priority-c system, the response time of a job of size x is stochastically bounded by*

$$T(x) \leq_{\text{st}} B_{<r}\left(W^{\leq r}_{\text{M/G/1/Priority-}c} + \frac{(4c+2)gkc^{r+1}}{c-1} + kx\right),$$

where c is the guardrail rank width, $r = \lfloor \log_c x \rfloor$ is the rank of the job, and $B_{<r}(w)$ is the length of a busy period comprising only jobs of rank $< r$ started by work w.

Proof. Let

- j be the tagged job,

- x be j's size and $r = \lfloor \log_c x \rfloor$ be j's rank,
- a_j and d_j be j's arrival and departure times, respectively, and
- s be the server to which j is dispatched.

The time at which job j departs, d_j, can be calculated as the time required for server s to complete the following work:

- relevant work already present at time a_j, namely $W_s^{\leq r}(a_j)$;
- plus all relevant work that arrives at s while j is in the system, namely $V_s^{<r}(t) - V_s^{<r}(a_j)$;
- plus j's size, namely x.

Let $t \geq a_j$ be an arbitrary time while j is in the system. Because each server runs at speed $1/k$, the amount of work that server s has completed by time t is $(t - a_j)/k$. Writing

$$Z_s(t) = W_s^{\leq r}(a_j) + (V_s^{<r}(t) - V_s^{<r}(a_j))$$

gives the following expression for j's departure time d_j:

$$d_j = \inf\left\{ t \,\middle|\, \frac{t - a_j}{k} \geq Z_s(t) + x \right\}. \tag{9}$$

To bound d_j, we first bound $Z_s(t)$. Let

$$\overline{Z}(t) = \frac{1}{k} \sum_{s'} Z_{s'}(t) \tag{10}$$

be the average value of $Z_{s'}(t)$ over all servers s', and let

$$Z_s^{\text{maxdiff}}(t) = \max_{s'}(Z_s(t) - Z_{s'}(t))$$

be the maximum difference between $Z_s(t)$ and $Z_{s'}(t)$ over all servers s'. Observe that

$$Z_s(t) \leq \overline{Z}(t) + Z_s^{\text{maxdiff}}(t).$$

Combining this with (9) gives a bound on d_j:

$$d_j \leq \inf\left\{ t \,\middle|\, \frac{t - a_j}{k} \geq \overline{Z}(t) + Z_s^{\text{maxdiff}}(t) + x \right\}. \tag{11}$$

To simplify (11), our next step is to bound $Z_s^{\text{maxdiff}}(t)$. We start by expanding $Z_s(t) - Z_{s'}(t)$:

$$Z_s(t) - Z_{s'}(t) = (W_s^{\leq r}(a_j) - W_{s'}^{\leq r}(a_j)) + (V_s^{<r}(t) - V_s^{<r}(a_j)) - (V_{s'}^{<r}(t) - V_{s'}^{<r}(a_j)). \tag{12}$$

We are left with an expression in terms of the rank $< r$ work dispatched to each server. We would like to turn this into an expression in terms of guardrail work counters, which will allow us to apply the constraints given by Definition 1. Consider the time interval (a_j, t). Job j is present at server s for the duration of the interval, so server s does not reset, implying

$$V_s^{<r}(t) - V_s^{<r}(a_j) = G_s^{<r}(t) - G_s^{<r}(a_j). \tag{13}$$

In contrast, server s' may reset during (a_j, t). When a reset occurs at some time t_{reset}, $G_{s'}^{<r}(t_{\text{reset}})$ decreases while $V_{s'}^{<r}(t_{\text{reset}})$ stays constant. Furthermore, $G_{s'}^{<r}(t')$ and $V_{s'}^{<r}(t')$ change in the same way at all other times t', so

$$V_{s'}^{<r}(t) - V_{s'}^{<r}(a_j) \geq G_{s'}^{<r}(t) - G_{s'}^{<r}(a_j). \tag{14}$$

Applying (13), (14), and Lemma 1 to (12) yields the bound

$$Z_s(t) - Z_{s'}(t) \leq \frac{2gc^{r+2}}{c - 1} + (G_s^{<r}(t) - G_s^{<r}(a_j)) - (G_{s'}^{<r}(t) - G_{s'}^{<r}(a_j)).$$

Because G-P is a guarded policy, we can apply the guardrail constraints from Definition 1:

$$\begin{aligned}
Z_s(t) - Z_{s'}(t) &\leq \frac{2gc^{r+2}}{c-1} + (G_s^{<r}(t) - G_{s'}^{<r}(t)) - (G_s^{<r}(a_j) - G_{s'}^{<r}(a_j)) \\
&= \frac{2gc^{r+2}}{c-1} + \sum_{q=-\infty}^{r-1} (G_s^q(t) - G_{s'}^q(t)) + (G_{s'}^q(a_j) - G_s^q(a_j)) \\
&\leq \frac{2gc^{r+2}}{c-1} + \sum_{q=-\infty}^{r-1} (gc^{q+1} + gc^{q+1}) \\
&= \frac{(2c+2)gc^{r+1}}{c-1}.
\end{aligned}$$

We have bounded $Z_s(t) - Z_{s'}(t)$ for arbitrary s' and hence bounded $Z_s^{\text{maxdiff}}(t)$. Substituting into (11) yields

$$d_j \leq \inf\left\{t \,\middle|\, \frac{t-a_j}{k} \geq \overline{Z}(t) + \frac{(2c+2)gc^{r+1}}{c-1} + x\right\}.$$

Recalling the definition of $\overline{Z}(t)$ from (10) gives us

$$d_j \leq \inf\left\{t \,\middle|\, t - a_j \geq W^{\leq r}(a_j) + (V^{<r}(t) - V^{<r}(a_j)) + \frac{(2c+2)gkc^{r+1}}{c-1} + kx\right\}.$$

Because the arrival process to the overall system is a Poisson process, we can rewrite this in terms of a "relevant" busy period, meaning one containing only jobs of rank $< r$:

$$d_j - a_j \leq B_{<r}\left(W^{\leq r}(a_j) + \frac{(2c+2)gkc^{r+1}}{c-1} + kx\right).$$

The Poisson arrival process also implies, by the PASTA property [30], that the amount of relevant work j sees on arrival, namely $W^{\leq r}(a_j)$, is drawn from the steady-state distribution, namely $W^{\leq r}$, so

$$T(x) \leq_{\text{st}} B_{<r}\left(W^{\leq r} + \frac{(2c+2)gkc^{r+1}}{c-1} + kx\right).$$

Applying Lemma 2 to $W^{\leq r}$ yields the desired bound. □

Remark 1. Note we can prove Lemmas 1 and 3 using only the following properties of resets:

- A server only resets when it is empty.
- When a server resets, its work counters do not increase.
- The guardrail constraints in Definition 1 continue to hold after each reset.

In particular, this means that resets are optional for proving our response time bounds, so the bounds hold even if the dispatcher chooses to omit some resets. This is helpful when implementing guarded dispatching policies in large systems (see Sections 6.1 and 6.2).

4.3.4 Proof of Theorem 1.

Theorem 1. *For any dispatching policy P, consider the policy G-P with tightness g. The expected response time for a job of size x under G-P/Priority-c is bounded by*

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-c}} \leq \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^r})(1-\rho_{c^{r+1}})} + \frac{(4c+2)gk\frac{c^{r+1}}{c-1} + kx}{(1-\rho_{c^r})},$$

where

- *$f_X(\cdot)$ is the probability density function of X,*

- *c is the guardrail rank width*
- *$r = \lfloor \log_c x \rfloor$ is the rank of a job of size x, and*
- *$\rho_y = \lambda \int_0^y t f_X(t)\, dt$ is the load due to jobs of size $\leq y$.*

Proof. Recall the conclusion of Lemma 3,

$$T(x) \leq B_{<r}\left(W^{\leq r}_{\text{M/G/1/Priority-}c} + \frac{(4c+2)gkc^{r+1}}{c-1} + kx\right). \tag{15}$$

Standard results on busy periods [13] state that

$$\mathbf{E}[B_{<r}(Y)] = \frac{\mathbf{E}[Y]}{1-\rho_{c^r}},$$

and standard results on the single-server Priority-c system[13] give the expected steady-state remaining rank $\leq r$ work:

$$\mathbf{E}[W^{\leq r}_{\text{M/G/1/Priority-}c}] = \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^{r+1}})}.$$

Taking expectations of (15) and applying these standard results yields the desired bound. □

4.4 Asymptotic Behavior of Guarded Policies

Theorem 2. *Consider a single-server SRPT system whose single server is k times as fast as each server in the load balancing system. For any dispatching policy P, consider the policy G-P with any constant tightness. Then for any size distribution X which is either (i) bounded or (ii) unbounded with tail having upper Matuszewska index[5] less than -2, the mean response times of G-P/SRPT, G-P/Priority-c, and (single-server) SRPT converge as load approaches capacity:*

$$\lim_{\rho\to 1} \frac{\mathbf{E}[T]^{\text{G-P/SRPT}}}{\mathbf{E}[T]^{\text{SRPT}}} = \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} = 1.$$

Proof. We start with the bound on the mean response time of G-P/Priority-c from Theorem 1:

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^r})(1-\rho_{c^{r+1}})} + \frac{(4c+2)gk\frac{c^{r+1}}{c-1} + kx}{(1-\rho_{c^r})}. \tag{16}$$

Note that the first term in this expression also appears in the expression for mean response time under single-server Priority-c [13]:

$$\mathbf{E}[T(x)]^{\text{Priority-}c} = \frac{\frac{\lambda}{2}\int_0^{c^{r+1}} t^2 f_X(t)dt}{(1-\rho_{c^r})(1-\rho_{c^{r+1}})} + \frac{x}{(1-\rho_{c^r})}.$$

Therefore, let us simplify (16):

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq \mathbf{E}[T(x)]^{\text{Priority-}c} + \frac{(4c+2)gk\frac{c^{r+1}}{c-1} + (k-1)x}{(1-\rho_{c^r})}. \tag{17}$$

We may simplify this bound by combining constant terms. Note that $c \leq 2$ and $x \geq c^r$. Let $m = 20gk + k - 1$. Then

$$mx \geq (4c+2)gkc^{r+1} + (k-1)(c-1)x.$$

Thus, we may simplify (17) further:

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq \mathbf{E}[T(x)]^{\text{Priority-}c} + \frac{mx}{(c-1)(1-\rho_{c^r})}.$$

[5]See Appendix A.

We want to relate this to something more similar to the mean response time under SRPT. A convenient policy for comparison with Priority-c that is similar enough to SRPT is Preemptive-Shortest-Job-First (PSJF), which prioritizes jobs according to their original size.

We now use Lemma 4 from Appendix B, which says that

$$\mathbf{E}[T]^{\text{Priority-}c} \leq (c + 2\sqrt{c-1})\mathbf{E}[T]^{\text{PSJF}}.$$

For brevity, let

$$b(c) = (c + 2\sqrt{c-1}).$$

Using Lemma 4, we find that

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq b(c)\mathbf{E}[T]^{\text{PSJF}} + \frac{mx}{(c-1)(1-\rho_{c^r})}.$$

Note that $x \geq c^r$, so $\rho_x \geq \rho_{c^r}$, so

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq b(c)\mathbf{E}[T]^{\text{PSJF}} + \frac{mx}{(c-1)(1-\rho_x)}.$$

Based on standard results on mean response time under PSJF and SRPT [13], we know that

$$\mathbf{E}[T(x)]^{\text{PSJF}} \leq \mathbf{E}[T(x)]^{\text{SRPT}} + \frac{x}{1-\rho_x}.$$

Let $m' = m + 4$. Because $c \leq 2$, we know $b(c) \cdot (c-1) \leq 4$. Thus,

$$\mathbf{E}[T(x)]^{\text{G-P/Priority-}c} \leq b(c)\mathbf{E}[T(x)]^{\text{SRPT}} + \frac{m'x}{(c-1)(1-\rho_x)}. \tag{18}$$

Next, we take the expectation of (18) over all job sizes x. To do so, we need to integrate

$$\int_0^\infty \frac{x}{1-\rho_x} f_X(x)dx,$$

where $f_X(\cdot)$ is the probability density function of X. To compute the integral, we make a change of variables from x to ρ_x, using the following facts:

$$\begin{aligned}\rho_x &= \int_0^x \lambda t f_X(t)dt \\ \frac{d\rho_x}{dx} &= \lambda x f_X(x) \\ \rho_0 &= 0 \\ \lim_{x\to\infty} \rho_x &= \rho.\end{aligned}$$

Given this change of variables, we compute

$$\int_0^\infty \frac{x}{1-\rho_x} f_X(x)dx = \int_0^\rho \frac{1}{\lambda(1-\rho_x)} d\rho_x = \frac{1}{\lambda} \ln\left(\frac{1}{1-\rho}\right).$$

Applying this to (18), we find that

$$\mathbf{E}[T]^{\text{G-P/Priority-}c} \leq b(c)\mathbf{E}[T]^{\text{SRPT}} + \frac{m'}{\lambda(c-1)} \ln\left(\frac{1}{1-\rho}\right).$$

Dividing through by $\mathbf{E}[T]^{\text{SRPT}}$, we find that

$$\frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} \leq b(c) + \frac{m' \ln \frac{1}{1-\rho}}{\lambda(c-1)\mathbf{E}[T]^{\text{SRPT}}}.$$

Plugging in the value of c in terms of ρ from (2),

$$\frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} \le b(c) + \frac{m' \cdot (\ln^2 \frac{1}{1-\rho} + \ln \frac{1}{1-\rho})}{\lambda \mathbf{E}[T]^{\text{SRPT}}}.$$

We now take the limit of the above ratio as $\rho \longrightarrow 1$. In this limit,

- $\ln \frac{1}{1-\rho}$ diverges,
- λ approaches $\mathbf{E}[X]$, and
- c approaches 1, so $b(c)$ also approaches 1:

$$\lim_{c \longrightarrow 1+} c + 2\sqrt{c-1} = 1.$$

Therefore, letting $m'' = 2m'/\mathbf{E}[X]$,

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} \le \lim_{\rho \to 1} \left(1 + \frac{m'' \ln^2 \frac{1}{1-\rho}}{\mathbf{E}[T]^{\text{SRPT}}} \right). \tag{19}$$

Recall now that we assume that X is either (i) bounded or (ii) unbounded with tail function having upper Matuszewska index[6] less than -2. In Lemma 5 in Appendix C, we use a result of Lin et al. [20] to show that in either case,

$$\lim_{\rho \to 1} \frac{\ln^2 \frac{1}{1-\rho}}{\mathbf{E}[T]^{\text{SRPT}}} = 0.$$

Applying this to (19), we find that

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{SRPT}}} \le 1. \tag{20}$$

SRPT yields optimal mean response time over all single-server policies [25], and a joint dispatching/scheduling policy can be emulated on a single server, so (20) is in fact an equality, as desired.

The optimality of SRPT's mean response time also implies that the mean response time under G-P/SRPT is no more than the mean response time under G-P/Priority-c. As a result,

$$\lim_{\rho \to 1} \frac{\mathbf{E}[T]^{\text{G-P/SRPT}}}{\mathbf{E}[T]^{\text{SRPT}}} = 1. \qquad \square$$

4.5 Optimality of Guarded Policies

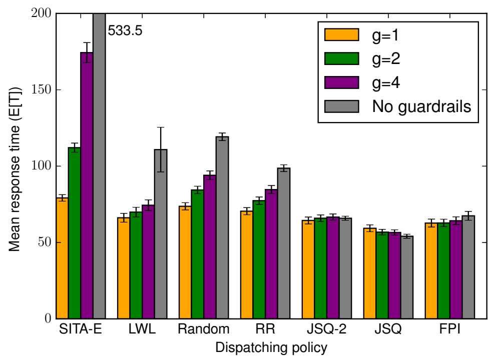
As a simple corollary of Theorem 2, we find that for any dispatching policy P, G-P/SRPT has optimal mean response time in the heavy traffic limit over all joint dispatching/scheduling policies.

Corollary 1. *For any dispatching policy P consider the policy G-P with any constant tightness. Consider any joint dispatching/scheduling policy P′/S′. Then for any size distribution X which is either (i) bounded or (ii) unbounded with tail having upper Matuszewska index*[6] *less than −2, the mean response times of G-P/SRPT and G-P/Priority-c are at least as small as the mean response time of P′/S′ as load approaches capacity:*

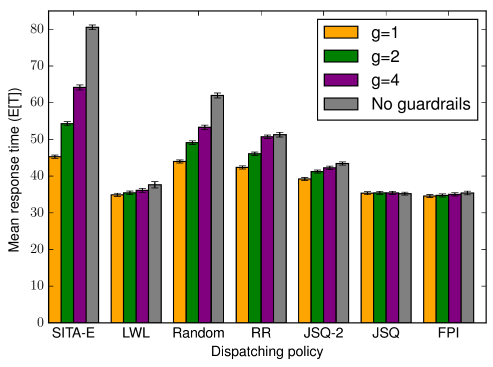
$$\lim_{\rho \to 1} \frac{\mathbf{E}[T]^{\text{G-P/SRPT}}}{\mathbf{E}[T]^{\text{P}'/\text{S}'}} = \frac{\mathbf{E}[T]^{\text{G-P/Priority-}c}}{\mathbf{E}[T]^{\text{P}'/\text{S}'}} \le 1.$$

Proof. SRPT has optimal mean response time among all single-server policies [25], and any joint dispatching/scheduling policy can be emulated on a single server, so $\mathbf{E}[T]^{\text{SRPT}} \le \mathbf{E}[T]^{\text{P}'/\text{S}'}$. The result thus follows from Theorem 2. □

[6]See Appendix A.



(a) $\rho = 0.98$



(b) $\rho = 0.80$

Fig. 4. At heavy load $\rho = 0.98$, adding guardrails significantly reduces the mean response times of SITA-E, LWL, Random, and RR, while leaving the mean response times of JSQ, JSQ-2, and FPI nearly unchanged. At more moderate load $\rho = 0.8$, adding guardrails significantly reduces the mean response times of SITA-E, Random, RR, and JSQ-2 while leaving the mean response times of LWL, JSQ, and FPI nearly unchanged. The smallest tightness, $g = 1$, shows the best performance for all policies except JSQ and FPI, where it doesn't really matter. Simulation uses $k = 10$ servers. Size distribution shown is Bounded Pareto with $\alpha = 1.5$ and range $[1, 10^6]$. $C^2 \sim 333$. 40 trials simulated, 95% confidence intervals shown.

5 SIMULATION

We have shown that in heavy traffic, adding guardrails to any dispatching policy gives it optimal mean response time. We now turn to investigating loads outside the heavy-traffic regime. While the mean response time upper bound in Theorem 1 holds for all loads, it is only tight in the heavy-traffic limit. We therefore focus on simulation.

We consider the following dispatching policies, each paired with SRPT scheduling at the servers:

Random The policy which dispatches each job to a uniformly random server.

Round-Robin (RR) The policy which dispatches each job to the server which least recently received a job.

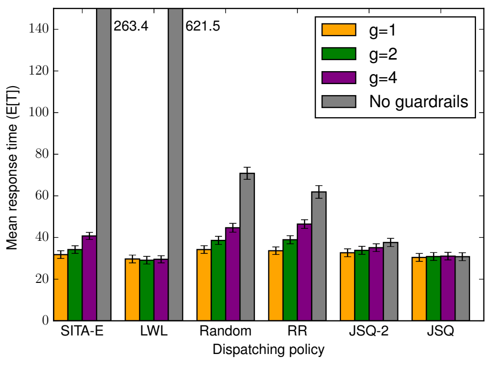
Least-Work-Left (LWL) The policy which dispatches each job to the server with the least remaining work.

Size-Interval-Task-Assignment (SITA) The policy which classifies jobs into size intervals (small, medium, large, etc.) and dispatches all small jobs to one server, all medium jobs to another server, etc. Specifically, we simulate **SITA-E**, the SITA policy which chooses the size intervals to equalize the expected load at each server.

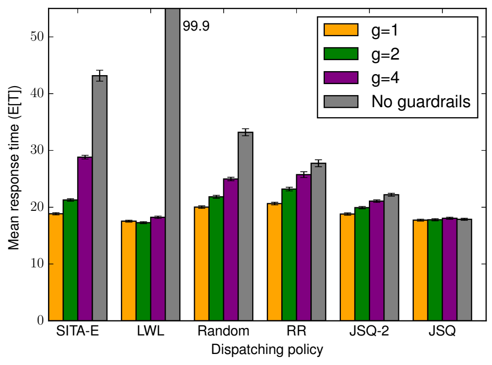
Join-Shortest-Queue (JSQ) The policy which dispatches each job to the server with the fewest jobs present.

Join-Shortest-Queue-*d* (JSQ-*d*) The policy which samples d uniformly random servers on each arrival and dispatches the job to the server with the fewest jobs present among those d. We focus on the $d = 2$ case.

First Policy Iteration (FPI) The first policy iteration heuristic, as described by Hyytiä et al. [17] in the setting of dispatching to SRPT servers. FPI dispatches each job to the server that would be optimal if all jobs thereafter were dispatched randomly. Hyytiä et al.'s derivation of the FPI policy assumes that the job size distribution is continuous, and specifically that two different jobs almost surely have different sizes. As a result, we only implement the FPI policy for the Bounded Pareto distribution, shown in Figure 4.



(a) $\rho = 0.98$



(b) $\rho = 0.80$

Fig. 5. At heavy load $\rho = 0.98$, adding guardrails significantly reduces the mean response time of SITA-E, LWL, Random, and RR, while leaving the mean response times of JSQ and JSQ-2 nearly unchanged. At more moderate load $\rho = 0.8$, adding guardrails significantly reduces the mean response times of SITA-E, LWL, Random, RR, and JSQ-2, while leaving the mean response time of JSQ nearly unchanged. The smallest tightness, $g = 1$, shows the best performance for all policies except JSQ, where it doesn't really matter. Simulation uses $k = 10$ servers. Size distribution shown is Bimodal with 99.95% size 1 jobs and 0.05% size 1000 jobs. $C^2 \sim 221$. 40 trials simulated, 95% confidence intervals shown.

5.1 Simulation Results

Figures 4 and 5 show the mean response time under all of the above dispatching policies with SRPT scheduling at the servers. We omit the FPI policy from Figure 5, because that simulation's job size distribution is not continuous, and Hyytiä et al. do not derive the FPI policy for such distributions [17]. We also show 95% confidence intervals for each mean response time. We consider two different job size distributions: a Bounded Pareto distribution (see Figure 4) and a Bimodal distribution (see Figure 5). In each case we show (a) very heavy traffic ($\rho = 0.98$) and (b) more moderate traffic ($\rho = 0.8$). We augment each dispatching policy with guardrails of varying tightness, $g = 1, 2$, and 4.

The high-level message seen in Figures 4 and 5 is that adding guardrails to dispatching policies can greatly reduce their response times, even at more moderate loads. Simple dispatching policies like Random and RR improve by $15 - 40\%$ when $\rho = 0.8$ and $30 - 50\%$ when $\rho = 0.98$, in the figures shown. Other policies, like LWL and SITA-E, show even more dramatic improvement for certain job size distributions. We find using tightness $g = 1$ is generally best.

The FPI heuristic of Hyytiä et al. [17] performs about equally well with or without guardrails. Figure 4 shows that adding guardrails to FPI yields a slight reduction in mean response time at $\rho = 0.98$, and has essentially no effect at $\rho = 0.8$. The FPI heuristic performs well in simulation, but its only theoretical guarantee is it outperforms Random. Applying guardrails guarantees optimal mean response time in heavy traffic, while maintaining or improving performance in simulation.

We observe that the JSQ dispatching policy performs well even without guardrails. In fact, guardrails can be seen as helping all the other dispatching policies to improve their performance to approach JSQ. We do not know of any guarantees on JSQ's performance with SRPT servers, even under heavy traffic, unless we add guardrails to JSQ. Figures 4 and 5 show that adding guardrails to JSQ does not affect its performance much.

In Appendix D, we simulate guarded policies under a variety of alternative system conditions. We simulate systems with more servers, systems with lighter load and systems with different job size distributions.

5.2 Simulation Discussion and Intuition

Recall from Section 2.1 the intuition behind guardrails: Guardrails force the dispatching policy to spread out small jobs across all of the servers. Guardrails thereby ensure that the maximum possible number of servers are working on the smallest jobs available.

Let us consider this intuition in light of each of the dispatching policies. SITA-E does the opposite of spreading small jobs: It clumps the smallest jobs onto the same server. Therefore, G-SITA-E shows a massive improvement. In particular, we want g to be as low as possible ($g = 1$), corresponding to the greatest guardrail control, to prevent SITA-E from doing what it was designed to do.

Random and RR are better at spreading jobs naturally, but they still make mistakes. In particular, Random and RR do not differentiate between jobs of different sizes and they do not observe the state of the servers, so they only spread out the small jobs by chance. As a result, G-Random and G-RR show sizable improvements. The tightest guardrails ($g = 1$) increase the spread of the small jobs the most, and hence show the most improvement.

LWL does not spread out the small jobs. In fact, one huge job at a server can keep away all of the small jobs for a long time. However, LWL is so efficient at using its servers that one does not experience its shortcomings unless both load and job size variability are high. Under those circumstances, LWL has very high mean response times. At load $\rho = 0.98$ with the Bimodal size distribution shown in Figure 5, LWL's mean response time is 7 times worse than that of Random. Guardrails are particularly effective in these situations where LWL fails because they force small jobs onto the servers with one large job. The tightest guardrails ($g = 1$) force small jobs onto those servers most aggressively and hence show the lowest mean response time in Figures 4 and 5.

5.3 Comparing Simulation to Analytical Bounds

In Theorem 1 we established an analytical upper bound on the mean response time of any guarded policy. This bound is tight in the limit as load $\rho \to 1$, and implies the heavy traffic optimality of any guarded policy.

However, this bound is not tight under the more moderate loads simulated in this section. Under the system conditions shown in Figure 4, at load $\rho = 0.8$ and guardrail tightness $g = 1$, Theorem 1 implies that any guarded policy has mean response time at most 350, and that at load $\rho = 0.98$ the mean response time is at most 700. The actual performance of guarded policies is much better than this, as shown in Figure 4. Tightening our bound is a potential direction for future research.

6 PRACTICAL CONSIDERATIONS

We now discuss several useful properties of guardrails that help when implementing them in practical systems. We also extend guardrails to cover a broader range of applications.

6.1 Robustness to Network Delays

Guardrails are relatively simple to implement: the dispatcher stores work counters for each rank and each server, increasing the appropriate work counter whenever it dispatches a job (see Algorithm 1). For the most part, the dispatcher does not need to monitor the precise state of each server. The only exception is that whenever a server becomes empty, the server *resets*, which decreases all of the dispatcher's work counters for that server. As we will explain shortly, this complicates the implementation, particularly in settings with network delays.

Fortunately, resets are optional for the purposes of heavy-traffic optimality (see Remark 1). However, resets are still desirable because they help decrease response time at lower loads. Specifically, resets ensure that a guarded policy is always allowed to dispatch jobs to empty servers. We thus do not want to ignore resets entirely.

To implement resets without the dispatcher needing to track the remaining work at each server at all times, servers can send "reset messages" to the dispatcher when they become empty. This works well so long as messages do not experience network delays, because our analysis (see Lemmas 1 and 3) assumes that servers only reset when they are empty, which might not be the case if a reset message is delayed.

In practice, reset messages may well experience network delays, To handle delays, the dispatcher should ignore reset messages from servers that might not be empty. One protocol for doing so is:

- The dispatcher stores, for each server s, a hash of all the job IDs sent to s.
- Each server s stores a hash of all the job IDs it has received.
- When a server becomes empty, it sends a reset message to the dispatcher which contains the currently stored hash.
- When the dispatcher receives a reset message from server s, it resets s if the reset message's hash matches the stored hash for s. Otherwise, the dispatcher ignores the reset message.

6.2 Multiple Dispatchers

Many large load balancing systems in practice have multiple dispatchers, each of which sends jobs to the same set of servers. Communication between the dispatchers may be limited, in which case they each have to make dispatching decisions independently. Fortunately, in systems with multiple dispatchers, it suffices to have each dispatcher independently implement guardrails. As explained below, we obtain the same theoretical guarantees for each of the following:

- A system with d dispatchers, each independently satisfying guardrails with tightness g.
- A system with a single dispatcher satisfying guardrails with tightness dg.

Guardrails thus guarantee heavy-traffic optimality for systems with any constant number of dispatchers.

To see why it suffices to implement guardrails separately for each dispatcher, consider a system with d dispatchers. Suppose each dispatcher separately keeps "local" guardrail work counters, which only track arrivals to that dispatcher, and implements guardrails with tightness g. We can also imagine what the "global" guardrail work counters, which track all arrivals at all dispatchers, would look like, even though there is no physical device storing them. We ask: given that the local counters have tightness g, what is the tightness of the global counters? Consider the local and global rank r guardrail work counters for two servers s and s'. Each dispatcher's local counter pair has difference at most gc^{r+1} (see Definition 1), and there are d dispatchers, so the global counter pair has difference at most dgc^{r+1}. This means the global counters stay within tightness dg.

Systems with multiple dispatchers tend to be large systems in which network delays are non-negligible. The reset protocol from Section 6.1 can be easily adapted to multiple dispatchers by having each server store a separate hash of job IDs for each dispatcher.

6.3 Scheduling Policies other than SRPT

We have shown that guarded dispatching policies provide theoretical guarantees and good empirical performance for load balancing systems using SRPT scheduling at the servers. However, in some settings it is impossible to use SRPT. For example, network hardware often allows scheduling using only finitely many priority classes, in which case SRPT can only be approximated [16, 23]. Systems may also choose a non-SRPT scheduling policy for other reasons, such as fairness concerns [27].

Guardrails are sometimes suitable even when the servers are using a scheduling policy other than SRPT. In particular, we can extend our theoretical guarantees to many preemptive size-based scheduling policies that favor small jobs. We have already proven such a guarantee for the Priority-c policy (see Theorem 2). Using bounds proved by Wierman et al. [28], one can extend our results to all policies in their SMART class, which includes Preemptive-Shortest-Job-First (PSJF) and Shortest-Processing-Time-Product (SPTP, also known as RS).

Guardrails can also provide guarantees for size-based policies with finitely many priority classes, which are used in some computer systems to approximate SRPT [16, 23]. In this setting, each priority class corresponds to an interval of job sizes. Here it is most natural to use a slightly modified version of guardrails: a guarded policy is one ensuring that for any class i, the maximum difference between two servers' class i work counters never exceeds the upper bound of class i's size interval. If the job size distribution is bounded, these modified guardrails guarantee a mean response time bound analogous to Theorem 1. This implies that in the heavy traffic limit, the system's performance approaches that of one large server using the same scheduling policy.

So far, we have only considered policies that use job size information to favor small jobs. This is the setting in which guardrails are most likely to be effective. We conjecture that guardrails might also be useful for servers using PS or Foreground-Background (FB) scheduling, as these policies also tend to favor small jobs, so they may benefit from spreading out small jobs across the servers.

6.4 Heterogeneous Server Speeds

We have thus far assumed that all servers in the system have the same speed, but this is not always the case. Fortunately, guardrails can be adapted to systems with heterogeneous server speeds. The key is to track each server's guardrail work counter $G_s(t)$ in units of time. That is, when we dispatch job of size x to a server with speed μ, we increase the server's guardrail work counter by x/μ. It is simple to generalize our response time bound in Theorem 1 to this setting by multiplying the bound's last term by $\mu_{\max}/\mu_{\min}$, where $\mu_{\min}$ and $\mu_{\max}$ are the minimal and maximal server speeds, respectively. This implies that any guarded policy paired with SRPT service is heavy-traffic optimal with heterogeneous servers.

7 CONCLUSION

We introduce *load balancing guardrails*, a technique for augmenting dispatching policies that ensures low response times in load balancing systems using SRPT scheduling at the servers. We prove that guardrails guarantee optimal mean response time in heavy traffic, and we show empirically that guardrails reduce mean response time across a range of loads. Moreover, guardrails are simple to implement and are a practical choice for large load balancing systems, including those with multiple dispatchers and network delays.

One direction for future work could address a limitation of guardrails: they require the dispatcher to know each job's exact size. Many computer systems only have access to noisy job size estimates or have no size information at all. When exact size information is not available, minimizing mean response time becomes much more complex, as it is not even clear what scheduling policy should be used at the servers. It is possible that a variation of guardrails could be used to create good dispatching policies when using the celebrated Gittins index scheduling policy [1, 11] at the servers.

Our analysis of guardrails constitutes the first closed-form mean response time bound for load balancing systems with general job size distribution and complex dispatching and scheduling policies. However, the bound is only tight in the heavy-traffic limit. Developing better analysis tools for the light traffic case remains an important open problem.

ACKNOWLEDGMENTS

We thank Gustavo de Veciana and the anonymous referees for their helpful comments. This research was supported by NSF-XPS-1629444, NSF-CSR-180341, and a 2018 Faculty Award from Microsoft. Additionally, Ziv Scully was supported by an ARCS Foundation scholarship and the NSF GRFP under Grant Nos. DGE-1745016 and DGE-125222.

REFERENCES

[1] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. 2009. On the Gittins index in the M/G/1 queue. *Queueing Systems* 63, 1 (2009), 437–458.

[2] E. Altman, U. Ayesta, and B. J. Prabhu. 2011. Load balancing in processor sharing systems. *Telecommunication Systems* 47, 1 (01 Jun 2011), 35–48. https://doi.org/10.1007/s11235-010-9300-8

[3] Nir Avrahami and Yossi Azar. 2003. Minimizing Total Flow Time and Total Completion Time with Immediate Dispatching. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '03)*. ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/777412.777415

[4] Eitan Bachmat and Hagit Sarfati. 2008. Analysis of Size Interval Task Assignment Policies. *SIGMETRICS Perform. Eval. Rev.* 36, 2 (Aug. 2008), 107–109. https://doi.org/10.1145/1453175.1453199

[5] T. Bonald, M. Jonckheere, and A. Proutiére. 2004. Insensitive Load Balancing. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '04/Performance '04)*. ACM, New York, NY, USA, 367–377. https://doi.org/10.1145/1005686.1005729

[6] F. Bonomi. 1990. On job assignment for a parallel system of processor sharing queues. *IEEE Trans. Comput.* 39, 7 (July 1990), 858–869. https://doi.org/10.1109/12.55688

[7] Maury Bramson, Yi Lu, and Balaji Prabhakar. 2012. Asymptotic independence of queues under randomized load balancing. *Queueing Systems* 71, 3 (01 Jul 2012), 247–292. https://doi.org/10.1007/s11134-012-9311-0

[8] Rodolpho G. de Siqueira and Daniel R. Figueiredo. 2017. A Control-based Load Balancing Algorithm with Flow Control for Dynamic and Heterogeneous Servers. In *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, Porto Alegre, RS, Brasil. http://portaldeconteudo.sbc.org.br/index.php/sbrc/article/view/2626

[9] Douglas G. Down and Rong Wu. 2006. Multi-layered round robin routing for parallel servers. *Queueing Systems* 53, 4 (01 Aug 2006), 177–188. https://doi.org/10.1007/s11134-006-7419-9

[10] Hanhua Feng, Vishal Misra, and Dan Rubenstein. 2005. Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems. *Performance Evaluation* 62, 1 (2005), 475 – 492. https://doi.org/10.1016/j.peva.2005.07.031 Performance 2005.

[11] John C. Gittins, Kevin D. Glazebrook, and Richard Weber. 2011. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons.

[12] Varun Gupta, Mor Harchol Balter, Karl Sigman, and Ward Whitt. 2007. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation* 64, 9 (2007), 1062–1081. https://doi.org/10.1016/j.peva.2007.06.012 Performance 2007.

[13] Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action* (1st ed.). Cambridge University Press, New York, NY, USA.

[14] Mor Harchol-Balter, Mark E. Crovella, and Cristina D. Murta. 1999. On Choosing a Task Assignment Policy for a Distributed Server System. *J. Parallel and Distrib. Comput.* 59, 2 (1999), 204–228. https://doi.org/10.1006/jpdc.1999.1577

[15] Mor Harchol-Balter, Alan Scheller-Wolf, and Andrew R. Young. 2009. Surprising Results on Task Assignment in Server Farms with High-variability Workloads. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '09)*. ACM, New York, NY, USA, 287–298. https://doi.org/10.1145/1555349.1555383

[16] Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. 2003. Size-based Scheduling to Improve Web Performance. *ACM Trans. Comput. Syst.* 21, 2 (May 2003), 207–233. https://doi.org/10.1145/762483.762486

[17] Esa Hyytiä, Aleksi Penttinen, and Samuli Aalto. 2012. Size- and state-aware dispatching problem with queue-specific job sizes. *European Journal of Operational Research* 217, 2 (2012), 357 – 370. https://doi.org/10.1016/j.ejor.2011.09.029

[18] Stefano Leonardi and Danny Raz. 2007. Approximating total flow time on parallel machines. *J. Comput. System Sci.* 73, 6 (2007), 875–891. https://doi.org/10.1016/j.jcss.2006.10.018

[19] Quan-Lin Li, John C. S. Lui, and Yang Wang. 2011. *A Matrix-Analytic Solution for Randomized Load Balancing Models with PH Service Times*. Springer Berlin Heidelberg, Berlin, Heidelberg, 240–253. https://doi.org/10.1007/978-3-642-25575-5_20

[20] Minghong Lin, Adam Wierman, and Bert Zwart. 2011. Heavy-traffic analysis of mean response time under Shortest Remaining Processing Time. *Performance Evaluation* (2011). https://doi.org/10.1016/j.peva.2011.06.001

[21] Zhen Liu and Rhonda Righter. 1998. Optimal Load Balancing on Distributed Homogeneous Unreliable Processors. *Operations Research* 46, 4 (1998), 563–573. https://doi.org/10.1287/opre.46.4.563 arXiv:https://pubsonline.informs.org/doi/pdf/10.1287/opre.46.4.563

[22] M. Mitzenmacher. 2001. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems* 12, 10 (Oct 2001), 1094–1104. https://doi.org/10.1109/71.963420

[23] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-driven Low-latency Transport Protocol Using Network Priorities. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. ACM, New York, NY, USA, 221–235. https://doi.org/10.1145/3230543.3230564

[24] Debankur Mukherjee, Sem Borst, Johan van Leeuwaarden, and Phil Whiting. 2016. Universality of Power-of-d Load Balancing Schemes. *SIGMETRICS Perform. Eval. Rev.* 44, 2 (Sept. 2016), 36–38. https://doi.org/10.1145/3003977.3003990

[25] Linus Schrage. 1968. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. *Operations Research* 16, 3 (1968), 687–690. http://www.jstor.org/stable/168596

[26] Richard R. Weber. 1978. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability* 15, 2 (1978), 406–413. https://doi.org/10.2307/3213411

[27] Adam Wierman and Mor Harchol-Balter. 2003. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 31. ACM, 238–249.

[28] Adam Wierman, Mor Harchol-Balter, and Takayuki Osogami. 2005. Nearly insensitive bounds on SMART scheduling. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33. ACM, 205–216.

[29] Wayne Winston. 1977. Optimality of the shortest line discipline. *Journal of Applied Probability* 14, 1 (1977), 181–189. https://doi.org/10.2307/3213271

[30] Ronald W. Wolff. 1982. Poisson arrivals see time averages. *Operations Research* 30, 2 (1982), 223–231.

[31] Xingyu Zhou, Jian Tan, and Ness Shroff. 2018. Flexible load balancing with multi-dimensional state-space collapse: Throughput and heavy-traffic delay optimality. *Performance Evaluation* 127-128 (2018), 176 – 193. https://doi.org/10.1016/j.peva.2018.10.003

A MATUSZEWSKA INDEX

The optimality results in this paper, such as Theorem 2, assume that the job size distribution X is not too heavy-tailed. Specifically, we assume that either X is bounded, or that the *upper Matuszewska index* of the tail of X is less than -2. This is slightly stronger than assuming that X has finite variance. The formal definition of the upper Matuszewska is the following.

Definition 2. Let f be a positive real function. The *upper Matuszewska index* of f, written $M(f)$, is the infimum over α such that there exists a constant C such that for all $\gamma > 1$,

$$\lim_{x\to\infty} \frac{f(\gamma x)}{f(x)} \leq C\gamma^{\alpha}.$$

Moreover, for all $\Gamma > 1$, the convergence $x \to \infty$ must be uniform in $\gamma \in [1, \Gamma]$.

The condition $M(\overline{F}_X) < -2$, where $\overline{F}_X$ is the tail of X, is intuitively close to saying that $F_X(x) \leq Cx^{-2-\epsilon}$ for some constant C and some $\epsilon > 0$. Roughly speaking, this means that X has a lighter tail than a Pareto distribution with $\alpha = 2$.

B LEMMA 4

Lemma 4. For any job size distribution, the mean response time of a single-server Priority-c system is no more than c times the mean response time of a single-server PSJF system:

$$\mathbf{E}[T]^{\text{Priority-}c} \leq (c + 2\sqrt{c-1})\mathbf{E}[T]^{\text{PSJF}}.$$

Proof. We will consider a new random variable, D, the delay due to a job. D is defined for scheduling policies that assign every job a fixed priority, like Priority-c and PSJF. For a given job j of size x, D_j is

- the amount j delays other jobs, namely x times the number of jobs with lower priority than j in the system when j arrives,

- plus the amount other jobs that arrived before j delay j, namely the total remaining size of jobs with higher priority than j in the system when j arrives,
- plus j's size.

Note that the response time of a job ℓ is equal to ℓ's size, plus the amount ℓ is delayed by jobs that arrived before ℓ, plus the amount l is delayed by jobs that arrive after ℓ. Each of those amounts of time is accounted for in the delay of exactly one job. As a result, the sum of the delays of the jobs in a busy period equals the sum of the response times of those jobs. Therefore, in steady state, mean response time and mean delay are equal:

$$\mathbf{E}[T] = \mathbf{E}[D].$$

Therefore, it suffices to show that

$$\mathbf{E}[D]^{\text{Priority-}c} \leq (c + 2\sqrt{c-1})\mathbf{E}[D]^{\text{PSJF}}.$$

Let us consider a pair of coupled systems receiving the same arrivals: A single-server system with PSJF scheduling, and a single-server system with Priority-c scheduling.

Note that PSJF and Priority-c both prioritize all jobs of lower ranks over all jobs of higher ranks. As a result, both coupled systems will server jobs of the same ranks at the same times, and will always have the same amount of remaining work of each rank.

Let us consider the expected delay due to a particular "tagged" job j, arriving to a steady-state system. The expected delay due to j is equal to each system's mean delay by the PASTA property [30]. Let j be a job of size x with rank $r = \log_c x$.

The delay due to j, D_j, is a summation over each job in the system at the moment j arrives. Let D_j^q be the delay caused by the interaction of j and jobs of rank q that are in the system when j arrives, including j's size in D_j^r. Then we can write D_j in terms of the D_j^qs:

$$D_j = \sum_{q=-\infty}^{\infty} D_j^q.$$

Therefore, it suffices to show for all ranks q that

$$\mathbf{E}[D_j^q]^{\text{Priority-}c} \leq (c + 2\sqrt{c-1})\mathbf{E}[D_j^q]^{\text{PSJF}}. \tag{21}$$

Let $X_{j'}$ denote the original size of a job j', and let $R_{j'}$ denote the remaining size of j'. Let J^q denote the set of jobs of rank q in the system at the time j arrives. Let N^q denote the number of jobs of rank q in the system at the time j arrives.

We now consider three cases: $q < r$, $q = r$, and $q > r$.

Case 1: $q < r$. Because $q < r$, all jobs in rank q have higher priority than j. As a result, under both PSJF and Priority-c, D_j^q is equal to the total remaining size of jobs of rank q:

$$D_j^q = \sum_{j' \in J^q} R_{j'}.$$

As noted above, this is equal in the two systems due to the coupling. This proves (21) in this case.

Case 2: $q = r$. Because Priority-c uses First-Come-First-Served scheduling within a rank, D_j^r in the Priority-c system is equal to the total remaining size of jobs of rank q:

$$D_j^{r\,(\text{Priority-}c)} = \sum_{j' \in J^{r\,(\text{Priority-}c)}} R_{j'}.$$

In contrast, D_j^r in the PSJF system is equal to the total remaining size of jobs of rank r with size at most x, plus x times the number of jobs of rank r with size more than x:

$$D_j^{r\,\text{(PSJF)}} = \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} \le x} R_{j'} \quad + \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} > x} x.$$

Noting that $x \ge c^r$ and the remaining size of any job of rank r is at most c^{r+1}, we can lower bound $D_j^{r\,\text{(PSJF)}}$:

$$\begin{aligned} D_j^{r\,\text{(PSJF)}} &\ge \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} \le x} R_{j'} \quad + \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} > x} c^r \\ &\ge \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} \le x} R_{j'} \quad + \sum_{j' \in J^{r\,\text{(PSJF)}} | X_{j'} > x} \frac{R_{j'}}{c} \\ &\ge \frac{1}{c} \sum_{j' \in J^{r\,\text{(PSJF)}}} R_{j'}. \end{aligned}$$

As noted above, $\sum_{j' \in J^q} R_{j'}$ is equal in both systems. As a result,

$$\mathbf{E}[D_j^q]^{\text{Priority-}c} \le c\mathbf{E}[D_j^q]^{\text{PSJF}},$$

which proves (21) in this case.

Case 3: $q > r$. Because $q > r$, all jobs in rank q have lower priority that j. As a result, in both systems,

$$D_j^q = xN^q.$$

Also, note that x is independent of N^q. Therefore, we simply need to show that

$$\mathbf{E}[N^q]^{\text{Priority-}c} \le (c + 2\sqrt{c-1})\mathbf{E}[N^q]^{\text{PSJF}}.$$

However, it is possible for there to be twice as many jobs of rank q in the Priority-c system as in the PSJF system, regardless of the value of c. In particular, there could be two rank q jobs in the Priority-c system and one rank q job in the PSJF system, if one of the rank q jobs in the Priority-c system has very little remaining size.

Let j^{old} be the oldest job of rank q in the Priority-c system at a given time. Because Priority-c serves jobs in FCFS order, only j^{old} has been processed, so only j^{old} can have remaining size under c^q. Therefore, we can bound the total remaining size of the rank q jobs in the Priority-c system:

$$\sum_{j' \in J^{q\,\text{(Priority-}c)}} R_{j'} \ge R_{j^{\text{old}}} + c^q(N^{q\,\text{(Priority-}c)} - 1).$$

Likewise, we can bound the total remaining size of the rank q jobs in the PSJF system:

$$\sum_{j' \in J^{q\,\text{(PSJF)}}} R_{j'} \le c^{q+1} N^{q\,\text{(PSJF)}}.$$

Using the fact that $\sum_{j' \in J^q} R_{j'}$ is equal in both systems gives us

$$R_{j^{\text{old}}} + c^q(N^{q\,\text{(Priority-}c)} - 1) \le c^{q+1} N^{q\,\text{(PSJF)}},$$

which rearranges to

$$N^{q\,\text{(Priority-}c)} \le cN^{q\,\text{(PSJF)}} + 1 - \frac{R_{j^{\text{old}}}}{c^q}.$$

This implies $N^{q(\text{Priority-}c)} \leq cN^{q(\text{PSJF})} + 1$, which allows us to relate the expected numbers of jobs in the systems:

$$\mathbf{E}[N^{q(\text{Priority-}c)}] \leq c\mathbf{E}[N^{q(\text{PSJF})}] + \mathbf{P}\{N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}\}. \tag{22}$$

Recall that $N^{q(\text{Priority-}c)}$ and $N^{q(\text{PSJF})}$ are both integers. This means that if $N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}$, then

$$N^{q(\text{Priority-}c)} \geq N^{q(\text{PSJF})} + 1.$$

As a result, if $N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}$, then

$$N^{q(\text{PSJF})} + 1 \leq cN^{q(\text{PSJF})} + 1 - \frac{R_{j^{\text{old}}}}{c^q},$$

meaning that

$$R_{j^{\text{old}}} \leq (c-1)c^q N^{q(\text{PSJF})}. \tag{23}$$

Therefore, we will show that either $\mathbf{P}\{N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}\}$ is small or $\mathbf{E}[N^{q(\text{PSJF})}]$ is large. In either case, (22) will imply the desired bound (21).

Let us condition on $R_{j^{\text{old}}}$. Because j is a Poisson arrival, j sees a time-average state of $R_{j^{\text{old}}}$. The (stochastically) smallest this distribution can be is the uniform distribution on $[0, c^q]$, because the original size of a rank q job is at least c^q. In particular, for any ℓ,

$$\mathbf{P}\{R_{j^{\text{old}}} < \ell\} \leq \frac{\ell}{c^q}.$$

Let ρ_q be the probability that j sees a rank q job in the system on arrival. Let m be the largest integer such that

$$\mathbf{P}\{N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}\} \geq m(c-1)\rho_q.$$

Then (23) implies

$$\mathbf{P}\{R_{j^{\text{old}}} \leq c^q(c-1)N^{q(\text{PSJF})}\} \geq m(c-1)\rho_q. \tag{24}$$

Regardless of the correlation between $N^{q(\text{PSJF})}$ and $R_{j^{\text{old}}}$, we must have

$$\mathbf{P}\{N^{q(\text{PSJF})} \geq m\} \geq (c-1)\rho_q. \tag{25}$$

This is because if $N^{q(\text{PSJF})} \leq m-1$ on a particular arrival, and also $N^{q(\text{Priority-}c)} > cN^{q(\text{PSJF})}$, then by (23),

$$R_{j^{\text{old}}} \leq (c-1)c^q(m-1). \tag{26}$$

But an arrival only sees job j^{old} in the system at all with probability ρ_q. Moreover, conditional on seeing a job j^{old} at all, the arrival observes (26) with probability at most $(c-1)(m-1)$, because every rank q job has size at least c^q. This means

$$\mathbf{P}\{R_{j^{\text{old}}} \leq (c-1)c^q(m-1)\} \leq (m-1)(c-1)\rho_q,$$

which together with (24) implies (25).

By a similar argument as (25),

$$\mathbf{P}\{N^{q(\text{PSJF})} \geq m-z\} \geq (z+1)(c-1)\rho_q$$

for any integer $z < m$.

In addition, if there is a job in the Priority-c system then there is a job in the PSJF system:

$$\mathbf{P}\{N^{q(\text{PSJF})} \geq 1\} \geq \rho_q.$$

We can combine these bounds on the probability of $N^{q\text{(PSJF)}}$ taking specific values to a derive a bound on its expectation:

$$\begin{aligned}\mathbf{E}[N^{q\text{(PSJF)}}] &\geq \left(\sum_{z=0}^{m-1}(m-z)(c-1)\rho_q\right) + (\rho_q - \rho_q m(c-1)) \\ &= \left(\left(\sum_{z=0}^{m-1}(m-z-1)(c-1)\right) + 1\right)\rho_q \\ &= \left(1 + \frac{m}{2}(m-1)(c-1)\right)\rho_q.\end{aligned}$$

We are now ready to show that either $\mathbf{P}\{N^{q\text{(Priority-}c\text{)}} > cN^{q\text{(PSJF)}}\}$ is small or $\mathbf{E}[N^{q\text{(PSJF)}}]$ is large by bounding their ratio for any value of m:

$$\begin{aligned}\frac{\mathbf{P}\{N^{q\text{(Priority-}c\text{)}} > cN^{q\text{(PSJF)}}\}}{\mathbf{E}[N^{q\text{(PSJF)}}]} &\leq \frac{(m+1)(c-1)\rho_q}{(1 + \frac{m}{2}(m-1)(c-1))\rho_q} \\ &= \frac{m+1}{\frac{1}{c-1} + \frac{m}{2}(m-1)}.\end{aligned}$$

This expression is maximized when

$$m = \sqrt{\frac{2c}{c-1}} - 1,$$

so

$$\begin{aligned}\frac{\mathbf{P}\{N^{q\text{(Priority-}c\text{)}} > cN^{q\text{(PSJF)}}\}}{\mathbf{E}[N^q]^{\text{(PSJF)}}} &\leq \frac{\sqrt{\frac{2c}{c-1}}}{\frac{1}{c-1} + (\sqrt{\frac{c}{c-1}} - \frac{1}{2})(\sqrt{\frac{2c}{c-1}} - 2)} \\ &= \frac{1}{\sqrt{\frac{2c}{c-1}} - \frac{3}{2}} \\ &\leq 2\sqrt{c-1}.\end{aligned}$$

where the final bound holds due to the fact that $1 < c \leq 2$, by the definition of c. Combining this result with (22) yields

$$\mathbf{E}[N^{q\text{(Priority-}c\text{)}}] \leq (c + 2\sqrt{c-1})\mathbf{E}[N^{q\text{(PSJF)}}],$$

which is (21), as desired. □

C LEMMA 5

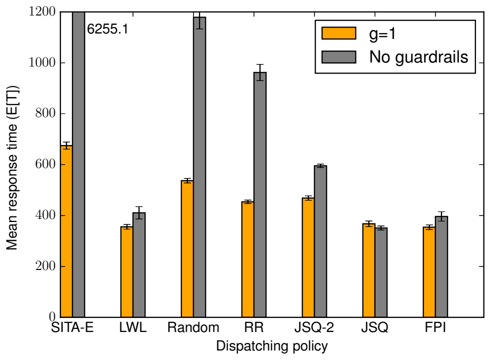
Lemma 5. For any size distribution X which is either (i) bounded or (ii) unbounded with tail having upper Matuszewska index[7] less than -2,

$$\lim_{\rho\to 1} \frac{\ln^2 \frac{1}{1-\rho}}{\mathbf{E}[T]^{\text{SRPT}}} = 0.$$

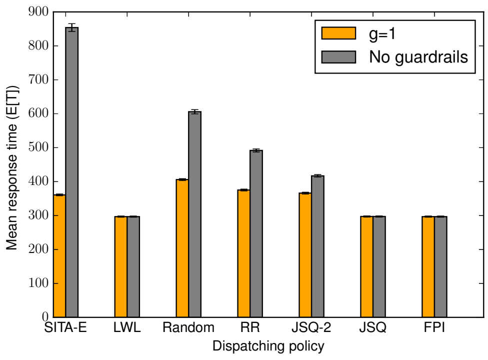
Proof. Lin et al. [20] show in their Theorem 1 that if X is bounded, then

$$\mathbf{E}[T]^{\text{SRPT}} = \theta\left(\frac{1}{1-\rho}\right),$$

[7]See Appendix A.



(a) $\rho = 0.98$



(b) $\rho = 0.80$

Fig. 6. Simulation with many servers: $k = 100$. Size distribution shown is Bounded Pareto with $\alpha = 1.5$ and range $[1, 10^6]$. $C^2 \sim 333$. 10 trials simulated, 95% confidence intervals shown.

proving case (i). They also show in their Theorem 2 that if the Matuszewska index of the tail of X is less than -2, then

$$\mathbf{E}[T]^{\text{SRPT}} = \theta\left(\frac{1}{(1-\rho)H^{-1}(\rho)}\right),$$

where $H^{-1}(\cdot)$ is the inverse of $H(x) = \rho_x/\rho$. Furthermore, in their proof of Theorem 2, they show that for any function $\phi(y)$ such that $\phi(y) = o(y^{\epsilon})$ for all $\epsilon > 0$,[8]

$$\lim_{\rho\to 1} \phi\left(\frac{1}{1-\rho}\right)(1-\rho)H^{-1}(\rho) = 0.$$

Applying this result with $\phi(y) = \ln^2(y)$ proves in case (ii). □

D ADDITIONAL SIMULATIONS

We include here some additional simulation results covering a wider range of cases than Section 5. We only show the tightest guardrails ($g = 1$), as those guardrails generally yield the lowest mean response times. We vary the parameters of the simulations in from Section 5 in three different ways:

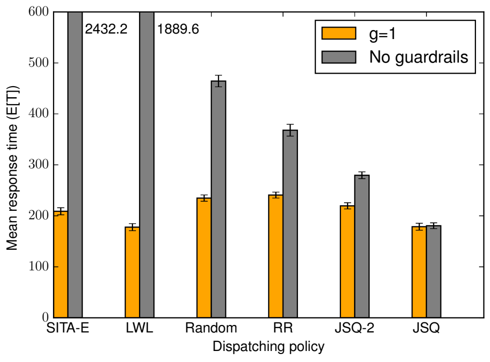
- adding many more servers (Figures 6 and 7),
- decreasing load (Figure 8), and
- varying the job size distribution (Figures 9 and 10).

In nearly every case guardrails improve or at least do not degrade mean response time of the underlying policy. In particular, as a general rule, G-LWL is nearly always tied for minimum mean response time among all dispatching policies simulated.

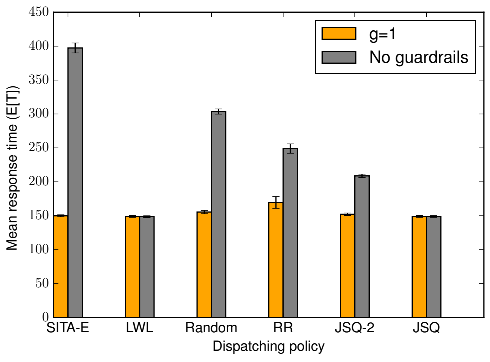
We omit the FPI heuristic from simulations involving the Bimodal job size distribution because Hyttiä et al. [17] only derive the policy for continuous job size distributions. We omit the FPI heuristic from the simulations of other job size distributions in Figures 9 and 10 due to lack of time.

Figures 6 and 7 show simulations with *many more servers*. Specifically, they use $k = 100$ servers, as opposed to $k = 10$ in other simulations. The only setup where guardrails degrade mean response time of the underlying policy is JSQ with Bounded Pareto job size distribution in heavy traffic, shown in Figure 6 (a), but G-LWL and G-FPI have performance on par with JSQ in that case.

[8]While Lin et al. [20] only mention this property for the specific case of $\phi(y) = \ln y$, their proof easily generalizes.

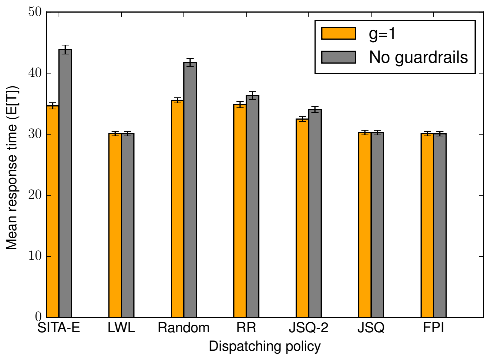


(a) $\rho = 0.98$

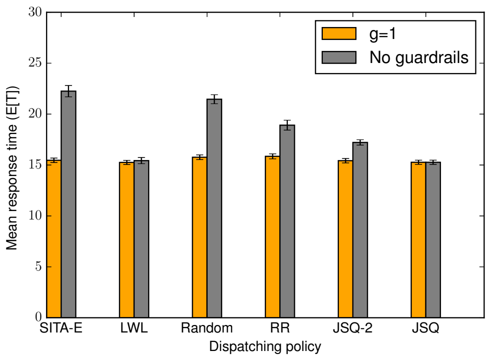


(b) $\rho = 0.80$

Fig. 7. Simulation with many servers: $k = 100$. Size distribution shown is Bimodal with 99.95% size 1 jobs and 0.05% size 1000 jobs. $C^2 \sim 221$. 10 trials simulated, 95% confidence intervals shown.



(a) Bounded Pareto job size distribution: $\alpha = 1.5$, range $[1, 10^6]$, $C^2 \sim 333$.

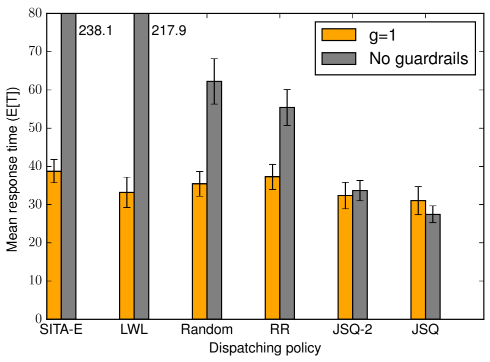


(b) Bimodal job size distribution: size 1 w.p. 99.95%, size 1000 w.p. 0.05%, $C^2 \sim 221$.

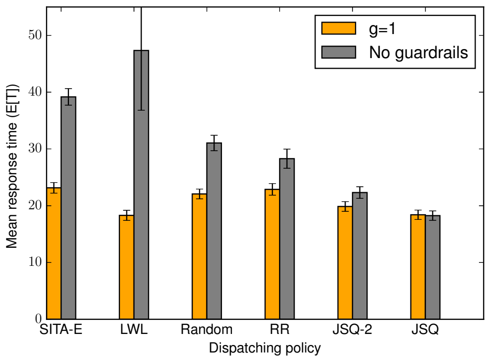
Fig. 8. Simulation with light traffic: $\rho = 0.5$. Simulation uses $k = 10$ servers. Two job size distributions shown. 10 trials simulated, 95% confidence intervals shown.

Figure 8 shows simulations with *light traffic*, specifically $\rho = 0.5$. The trends are largely the same as those in Section 5, but the differences in mean response time are smaller. This is to be expected because a large fraction of jobs experience no delay. In fact, the mean response time is nearly equal to the mean service time for many of the dispatching policies ($E[T] \approx 30$ in (a), $E[T] \approx 15$ in (b)). Guardrails are particularly effective in the Bimodal case shown in (b), dispatching nearly every small job to a server with no other small jobs. In addition to the loads shown, we have also simulated a range of loads from $\rho = 0.2$ to $\rho = 0.9975$. The trends are consistent across all loads, but the differences are less pronounced at lower loads.

Figures 9 and 10 show simulations with *different job size distributions*. We specifically simulate with Hyperexponential and Exponential job size distributions, representing another high-variance distribution and a low-variance distribution, respectively. In the Hyperexponential case shown in Figure 9, LWL performs particularly poorly without guardrails, similar to the Bimodal case. Roughly speaking, this is because there again are two types of jobs, though each has an exponential distribution instead of a deterministic one, and a job of the large type can cause many jobs of the

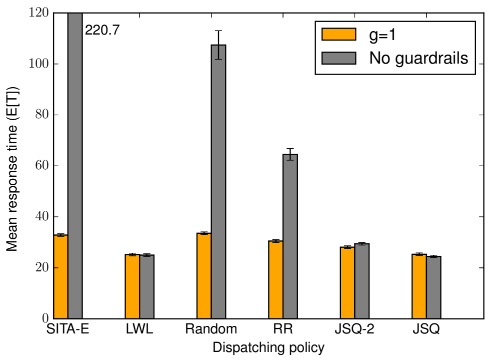


(a) $\rho = 0.98$

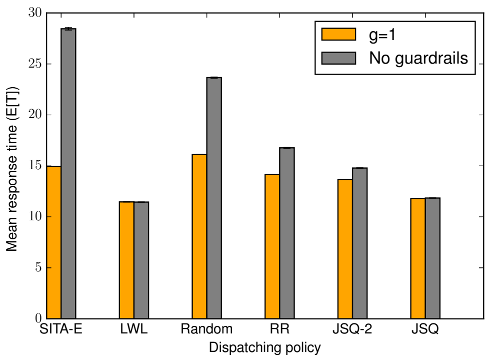


(b) $\rho = 0.80$

Fig. 9. Simulation with different size distribution: Hyperexponential with mean ~ 1.5 and $C^2 \sim 444$. Simulation uses $k = 10$ servers. 10 trials simulated, 95% confidence intervals shown.



(a) $\rho = 0.98$



(b) $\rho = 0.80$

Fig. 10. Simulation with different size distribution: Exponential with mean 1. Simulation uses $k = 10$ servers. 10 trials simulated, 95% confidence intervals shown.

small type to be dispatched to a single server. But again, guardrails effectively mitigate this problem, although JSQ slightly outperforms G-LWL in very heavy traffic. In the Exponential case, LWL without guardrails performs well already, but adding guardrails does not degrade its performance.

Received February 2019; revised March 2019; accepted April 2019