SPLIT: SymPathy for Large jobs Improves Tail latency

ZHOUZI LI, Carnegie Mellon University, United States
MOR HARCHOL-BALTER, Carnegie Mellon University, United States
ALAN SCHELLER-WOLF, Carnegie Mellon University, United States

We study the asymptotic response time tail in the M/G/n multi-server queue with heavy-tailed (regularly varying) job sizes, a setting representative of modern computing workloads. For single-server systems, tail optimization is well understood: under heavy-tailed job sizes, policies such as SRPT that strictly prioritize short jobs are strongly tail optimal, and giving any priority to large jobs is harmful. For multi-server systems, the question has been almost entirely open.

This paper gives the first strongly tail-optimal scheduling policies for the M/G/n queue with heavy-tailed job sizes. Our central finding is that the multi-server case is intrinsically different from the single-server case: giving a small amount of "sympathy" to large jobs is essential for strong tail optimality. We establish strong (or arbitrarily close to strong) tail optimality across the full stability region, both with and without knowledge of job sizes.

ACM Reference Format:
Zhouzi Li, Mor Harchol-Balter, and Alan Scheller-Wolf. 2026. SPLIT: SymPathy for Large jobs Improves Tail latency. 1, 1 (October 2026), 33 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

arXiv:2605.13749v3 [cs.PF] 18 Oct 2026

1 Introduction

Optimal scheduling has been an important topic in the performance modeling community for many decades. In the simplest setting, where there is just a single server and the goal is to minimize mean response time, the optimal scheduling policy is well understood: at all times, preemptively serve the job with the Shortest Remaining Processing Time (SRPT) [31]. The intuition is that a single large job occupying the server forces many smaller jobs to wait, inflating the response time of all of them; SRPT avoids this by always preempting in favor of shorter jobs. This intuition is so fundamental that prioritizing short jobs has become almost synonymous with optimal scheduling.

However, in addition to optimizing the mean response time, another important question is that of *optimal tail scheduling*. In many modern computing systems, the tail of the response time distribution matters at least as much as the mean: Metrics such as the 99th or 99.9th percentile of response time, often codified as service-level objectives (SLOs), directly govern user experience.

This paper develops the first tail optimal scheduling policies for multiserver systems with heavy-tailed job sizes. We formally define tail optimality in Section 1.1, briefly survey existing tail optimization results in Section 1.2, and explain why the multi-server case remains open in Section 1.3. We then present the main intuition behind our approach in Section 1.4.

Authors' Contact Information: Zhouzi Li, zhouzil@andrew.cmu.edu, Carnegie Mellon University, Computer Science Department, Pittsburgh, Pennsylvania, United States; Mor Harchol-Balter, harchol@cs.cmu.edu, Carnegie Mellon University, Computer Science Department, Pittsburgh, Pennsylvania, United States; Alan Scheller-Wolf, awolf@andrew.cmu.edu, Carnegie Mellon University, Tepper School of Business, Pittsburgh, Pennsylvania, United States.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM XXXX-XXXX/2026/10-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.1 Definition of tail optimality

A common theoretical formalization of tail optimization studies the asymptotic decay of $\mathbb{P}[T > t]$ as $t \to \infty$ [11, 23, 27, 37–39]. As argued in [39], there are several reasons to focus on this asymptotic regime: (i) optimizing $\mathbb{P}[T > t]$ for any single value of t is seldom the sole design objective; rather, one hopes to achieve low $\mathbb{P}[T > t]$ across a range of t; (ii) practical SLOs relate to high-quantile response times, which correspond to large values of t; and (iii) optimizing $\mathbb{P}[T > t]$ for fixed finite t appears to be theoretically intractable, whereas the asymptotic $t \to \infty$ limit has seen promising recent progress.

To make the notion of optimality precise, we say a scheduling policy π is *weakly tail optimal* if there exists a constant $c \geq 1$ such that

$$\sup_{\pi'} \limsup_{t \to \infty} \frac{\mathbb{P}[T^{\pi} > t]}{\mathbb{P}[T^{\pi'} > t]} = c,$$

and *strongly tail optimal* if additionally $c = 1$.

1.2 Brief summary of tail optimization results

For *single-server* systems, tail optimization is by now well understood, though the answer depends critically on whether the job size distribution is heavy-tailed or light-tailed. As shown by [37], no single policy can be weakly tail optimal for both heavy-tailed and light-tailed distributions simultaneously; the two settings require fundamentally different scheduling strategies.

Heavy-tailed (regularly varying) job size distributions are particularly important because they are widely observed in real-world computer workloads, including web file sizes [13], Unix process lifetimes [22], and modern cloud and datacenter job sizes [35]. Under heavy-tailed job sizes, a number of policies, including SRPT, Processor Sharing (PS), Foreground-Background (FB), and SMART, have been shown to be strongly tail optimal [8, 11, 19, 27, 34]. Together with the lower bound in [37, 39], these results completely characterize single-server tail optimality under heavy tails. Under light-tailed job sizes, the story is different: SRPT has the worst tail decay rate, while FCFS is weakly but not strongly tail optimal [11, 18]; the strongly optimal policy Boost was proposed recently in [39]. We defer a more detailed discussion of the light-tailed case to Section 2.3.

Table 1 summarizes the state of affairs for single-server systems.

Table 1. Response time tail optimality results for single-server (M/G/1) systems.

	Light-tailed Sizes	**Heavy-tailed Sizes**
Weakly optimal	FCFS [11], Nudge [18]	
Strongly optimal (known sizes)	Boost [39]	SRPT [27], SMART [27]
Strongly optimal (unknown sizes)	γ-Gittins [23]	PS [8, 19], FB [27], Gittins [34]

1.3 Tail optimality is open for multi-server systems

The *multi-server* setting, by contrast, remains almost entirely open. The only prior result on tail-optimal scheduling in the M/G/n is due to [38], which studies light-tailed job sizes and shows that Boost is strongly tail optimal in the heavy-traffic regime; however, the performance of Boost degrades substantially outside of heavy traffic. We discuss this work in detail in Section 2.3. To the best of our knowledge, no existing policy has been proved to be strongly – or even weakly – tail optimal for multi-server systems with heavy-tailed job sizes, which is the setting most representative of real-world systems and workloads. This paper is the first to fill this gap. Table 2 summarizes the known results for multi-server systems, including the contributions of this paper.

Table 2. Tail optimality results for multi-server (M/G/n) systems.
*In the case when system load $\rho < \frac{n-1}{n}$.
†Arbitrarily close to strongly optimal.
+Based on Approximation 1, as used in [4, 5, 20].

	Light-tailed Sizes	**Heavy-tailed Sizes**
Weakly optimal		SRPT-n [**This paper**]
Strongly optimal (known sizes)	Boost [38] (heavy traffic only)	SPLIT * [**This paper**] Thresh-SPLIT(d)† [**This paper**]
Strongly optimal (unknown sizes)		TAGS-SPLIT(d)†+ [**This paper**]

We study tail-optimal scheduling in the M/G/n with heavy-tailed (regularly varying) job sizes. Our central finding is that optimal scheduling for multi-server systems is *intrinsically different* from optimal scheduling for single-server systems. On the one hand, we prove that SRPT-n (where the n servers at all times serve the n jobs with the shortest remaining processing time) is still *weakly* tail optimal (Theorem A.1), but it is not *strongly* tail optimal (Theorem B.1). On the other hand, we find that achieving *strong* tail optimality requires giving some "sympathy" to large jobs, a strategy that would be harmful in a single-server system.

1.4 Intuition behind our approach SPLIT

To understand why multi-server systems require a different approach, it is helpful to contrast the single-server and multi-server cases.

In a *single-server* system, the main concern is preventing small jobs from being stuck behind large ones. Under heavy-tailed job sizes, being blocked by a single large job can be catastrophic for the many small jobs waiting behind it. One might consider giving large jobs some help to improve the response time tail, since it is the large jobs that dominate the tail. However, any service given to a large job comes directly at the expense of small jobs waiting behind it: There is no way to help large jobs without hurting small ones. This is roughly why SRPT, which strictly prioritizes short jobs, is strongly tail optimal under heavy tails [27]. Even under light-tailed job sizes, where the optimal strategies are completely different and FCFS is weakly optimal, policies like Nudge [18] and Boost [39] still rely on giving short jobs extra priority over large ones to improve upon FCFS.

In a *multi-server* system, however, the situation is fundamentally different. Even if a large job occupies one server, the remaining servers can continue to process small jobs without interruption. This means we may be able to afford to give large jobs some sympathy, dedicating service capacity to them, without significantly hurting small jobs in the process. Since it is the large jobs that dominate the response time tail, this sympathy could improve tail performance. Moreover, an added perk of giving large jobs some priority is improved *packing*: Under SRPT-n, at the end of a busy period, only a few large jobs remain, resulting in underutilization of the servers (see [16] for detailed discussion). This underutilization can potentially be mitigated by finishing off some of these larger jobs early on. We discuss the effect of packing in detail in Section 8.2.

These ideas motivate our new policy SPLIT (Sympathy for Large jobs Improves Tail latency). Consider first the case where system load $\rho < \frac{n-1}{n}$. Here the system can afford to sacrifice one entire server to serve a large job, while the remaining servers continue to drain small jobs. In this regime, we design the SPLIT policy: The policy splits the n servers into two groups, where $n-1$ servers run SRPT-$(n-1)$, ensuring that small jobs are processed efficiently, while the remaining one server runs a non-preemptive Largest-Job-First (LJF) policy, giving large jobs dedicated service. We prove that SPLIT is strongly tail optimal for any load $\rho < \frac{n-1}{n}$ (Section 4).

When $\rho \geq \frac{n-1}{n}$, dedicating an entire server to large jobs is no longer feasible: Imagine a very large job occupying a server. During this long period of time, all arrivals must be served by the remaining $n-1$ servers, and the system becomes temporarily unstable, which inflates the response time of all small jobs arriving during this period of time. To handle this regime, we design Thresh-SPLIT(d), which uses a size threshold d to control the division of work. Jobs smaller than d are routed to the $n-1$ servers, which are guaranteed (by the choice of d) a load strictly below 1, while jobs larger than d are handled by one server running SRPT-1. By choosing d appropriately, Thresh-SPLIT(d) can be made arbitrarily close to strongly tail optimal under any load $\rho < 1$ (Section 5).

Finally, in Section 6, we consider the setting where exact job sizes are unknown. Motivated by the Task Assignment by Guessing Size (TAGS) policy in [20], we generalize the Thresh-SPLIT(d) policy to the TAGS-SPLIT(d) policy. We show that by choosing the threshold d appropriately, TAGS-SPLIT(d) can also be made arbitrarily close to strongly tail optimal under all loads $\rho < 1$.

1.5 Contributions

Our main contributions are as follows.

- We prove that SRPT-n is weakly tail optimal, but not strongly tail optimal, for the M/G/n queue with heavy-tailed job sizes. (Appendices A and B)
- We introduce SPLIT, the first scheduling policy that is strongly tail optimal for the M/G/n queue with heavy-tailed job sizes. SPLIT achieves strong tail optimality when $\rho < \frac{n-1}{n}$, and empirically outperforms SRPT-n consistently across high response times. (Section 4)
- We introduce Thresh-SPLIT(d), a parameterized family of policies. By choosing the threshold d appropriately, Thresh-SPLIT(d) can be made arbitrarily close to strongly tail optimal under any load $\rho < 1$. (Section 5)
- We generalize Thresh-SPLIT(d) to TAGS-SPLIT(d) for the setting where exact job sizes are unknown to the scheduler. Relying on a standard approximation in the TAGS literature (Approximation 1), TAGS-SPLIT(d) can likewise be made arbitrarily close to strongly tail optimal under any load $\rho < 1$. (Section 6)

2 Prior Work

We now review the most relevant prior work, organized into three areas: scheduling policies that optimize mean response time (Section 2.1), analysis of tail behavior under various scheduling disciplines (Section 2.2), and policies designed to optimize tail latency (Section 2.3).

2.1 Scheduling to optimize mean response time

In the single-server M/G/1 queue, SRPT is known to minimize mean response time [31]. When exact job sizes are unknown, the Gittins index policy is mean-optimal [1, 33]. In the case of heavy traffic, these single-server results extend to the multi-server M/G/n; specifically, SRPT-n and M-Gittins are asymptotically mean-optimal in heavy traffic [17, 32].

Recently, Grosof and Hurtado-Lange [16] introduced the SEK policy (see Section 7 for more discussion), which is the only prior work observing that giving some priority to large jobs can benefit mean response time. They prove that SEK just slightly outperforms SRPT-n for mean response time at all loads.

Our work differs fundamentally in objective: we optimize the *tail* of the response time rather than the mean. As established by Wierman and Zwart [37], tail optimization is a fundamentally different problem from mean optimization, requiring qualitatively different scheduling strategies.

2.2 Analysis of tail behavior

A substantial body of work has analyzed the *tail* of the response time distribution under various scheduling policies. These works focus on characterizing tail behavior rather than establishing optimality results.

Single-server results. In the M/G/1 with regularly varying service times of index $(-\alpha)$, the response time tail depends critically on the scheduling discipline. FCFS yields a *heavier* tail of index $-(\alpha-1)$, since a single large job delays all subsequent arrivals [7, 11]. Disciplines that protect small jobs from such blocking, including PS, FB, and SRPT, instead preserve the original service time index $(-\alpha)$ [8, 19, 26, 27, 40], and the full range of achievable indices is characterized in [10]. Under light-tailed job sizes the ordering reverses: the response time tail decays fastest under FCFS [2], while SRPT produces the *worst* tail rather than the best [28].

Multi-server results. The multi-server case is far less understood. To the best of our knowledge, all existing tail-analytic results concern FCFS, covering the two-server queue and its heterogeneous and low-load variants [6, 9, 14, 15].

Notably, all of these results analyze specific policies without addressing optimality, and the tail behavior of non-FCFS policies in multi-server systems remains entirely unexplored prior to the current paper.

2.3 Scheduling to optimize tail latency

In the previous subsection, we covered work which analyzes the tail behavior of various specific policies, that work did not identify which policies are *optimal* in their response time tail. We now discuss prior work on tail-optimality.

Single-server results. For *heavy-tailed* job sizes, the setting that is the focus of this paper, single-server tail optimality is already characterized in Section 1.2 (see Table 1), so we do not repeat it here. The *light-tailed* case was only mentioned briefly in the introduction, so we expand on it now. Boxma and Zwart [11] showed that FCFS is weakly tail optimal. Grosof et al. [18] introduced Nudge, which stochastically improves upon FCFS, demonstrating that FCFS is not strongly tail optimal. Van Houdt [36] and Charlet and Van Houdt [12] built upon Nudge, further improving tail performance. Only recently, Yu and Scully [39] proposed Boost, the first policy achieving *strong* tail optimality under light-tailed distributions. Harlev et al. [23] extended strong tail optimality to unknown job sizes via a Gittins-based policy.

Importantly, none of these policies, whether designed for heavy-tailed or light-tailed distributions, prioritizes large jobs. The optimal policies for heavy-tailed distributions (e.g., SRPT, SMART) all prioritize short jobs. Even for light-tailed distributions where FCFS is weakly optimal, Nudge and Boost both give short jobs extra priority over large ones to improve upon FCFS.

Multi-server results. While the landscape of tail optimality is mostly clear in single-server systems, the multi-server setting remains almost entirely open. The only prior work on tail-optimal scheduling in multi-server systems is Yu et al. [38], which studies *light-tailed* job sizes and shows that Boost is strongly tail optimal in heavy traffic. However, outside of this regime, the tail performance of Boost degrades significantly. Interestingly, the same paper proposes a heuristic called CombinedBoost that empirically achieves strong tail performance even outside of heavy traffic. In CombinedBoost, large jobs are given some priority for a better packing effect. However, no theoretical analysis is provided for CombinedBoost; all evaluations are numerical.

To the best of our knowledge, no existing policy has been proved to be strongly, or even weakly, tail optimal for multi-server systems with heavy-tailed job sizes: This paper is the first to do so.

3 Problem Setting and Notation

In this section, we formally describe the system model, define the key notation, and establish preliminary lower bounds on the response time tail that serve as benchmarks for our policies.

3.1 Problem Setting

System model. We consider an M/G/n queueing system with n identical servers. Jobs arrive according to a Poisson process with rate λ, and each job has a random size drawn independently from a distribution S.

Heavy-tailed job sizes. Following [37], we focus on *regularly varying* job size distributions, i.e., distributions with Pareto-like tails. Formally, a job size distribution S is regularly varying with index $\alpha > 0$ if for any $k > 0$,

$$\lim_{t\to\infty} \frac{\mathbb{P}[S > kt]}{\mathbb{P}[S > t]} = k^{-\alpha}. \tag{1}$$

We assume $\alpha > 1$ so that the mean job size $\mathbf{E}[S]$ is finite.

Preemptible jobs. Jobs are preempt-resume, meaning a preempted job can later continue from where it left off with no loss of work. Preemptibility is essential in the heavy-tailed setting: [3] shows that no non-preemptive scheduling discipline can prevent large delays from building up under heavy-tailed job sizes (see also [11] for a survey).

Known job sizes. We initially assume that job sizes are known to the scheduler upon arrival. In Section 6, we relax this assumption and introduce TAGS-SPLIT(d).

Load and resource requirement. We define the *load* of the system as $\rho := \frac{1}{n}\lambda\mathbf{E}[S]$ and the *resource requirement* as $r := \lambda\mathbf{E}[S] = n\rho$; here r represents the minimum number of servers needed to keep the system stable [21]. We will also need the corresponding notation for the resource requirement contributed by jobs with size larger than x:

$$r_{>x} := \lambda \cdot \mathbb{P}[S > x] \cdot \mathbf{E}[S \mid S > x] = \lambda\mathbf{E}[S \cdot \mathbf{1}\{S > x\}]. \tag{2}$$

Objective. Our goal is to minimize the asymptotic tail of the response time distribution, in the sense of weak and strong tail optimality defined in Section 1.1.

3.2 Preliminaries

In this subsection, we first recall the known optimal tail for single-server systems with heavy-tailed job sizes. Then we use the single-server result to derive two lower bounds on the tail of any multi-server policy, each of which is tighter for a different load range.

Single-server optimal tail. For a single-server system with heavy-tailed job sizes, prior works [11, 37, 39] have established that the optimal tail decay rate (achieved by policies like SRPT [27]) is

$$\lim_{t\to\infty} \frac{\mathbb{P}[T > t]}{\mathbb{P}[S > (1-\rho)t]} = 1. \tag{3}$$

Lower bound 1: Super server. Consider a single "super server" with speed n. Since a super server can essentially mimic the behavior of n unit-speed servers, it strictly dominates any system of n unit-speed servers. Therefore, for any multi-server policy π,

$$\mathbb{P}[T^{\pi} > t] \geq \mathbb{P}[T^{\text{super}} > t], \tag{4}$$

where T^{super} denotes the response time under the optimal policy for the super server. Note that the load ρ is the same in both the M/G/n and the super server, since the super server has speed n and

effectively sees jobs of size S/n. Applying (3) with effective job size S/n yields

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\pi} > t\right]}{\mathbb{P}\left[\frac{S}{n} > (1-\rho)t\right]} \geq \lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{super}} > t\right]}{\mathbb{P}\left[\frac{S}{n} > (1-\rho)t\right]} = 1. \tag{5}$$

Lower bound 2: Job size. Since the response time is at least the job size, for any multi-server policy π and any $t > 0$,

$$\frac{\mathbb{P}\left[T^{\pi} > t\right]}{\mathbb{P}\left[S > t\right]} \geq 1. \tag{6}$$

Comparing the two lower bounds. Lower bound 1 from (5) has a larger denominator (and is therefore tighter) when $\rho > \frac{n-1}{n}$. Lower bound 2 from (6) is tighter when $\rho < \frac{n-1}{n}$.

4 SPLIT

In this section, we introduce the SPLIT (SymPathy for Large jobs Improves Tail latency) policy, which is strongly tail optimal when the load ρ satisfies $\rho < \frac{n-1}{n}$. For the case when $\rho \geq \frac{n-1}{n}$, we will show a different policy in Section 5.

4.1 Policy SPLIT Definition

We now define the SPLIT policy. See Figure 1 for an illustration.

Definition 4.1 (SPLIT). The policy SPLIT splits n servers into two groups: The first $n-1$ servers process jobs according to SRPT-$(n-1)$ and are called the *SRPT servers*, and the last one server serves according to Non-preemptive Largest Job First (LJF) and is called the *LJF server*. Specifically, whenever the LJF server is idle, it fetches the job with the largest (*original*) size in the system (either in the queue or being served by the SRPT servers) and serves it. If there are no jobs in the system, the LJF server will serve the next arriving job. At every moment of time, among the rest of the jobs, the $n-1$ jobs with the smallest *remaining size* run on the SRPT servers.

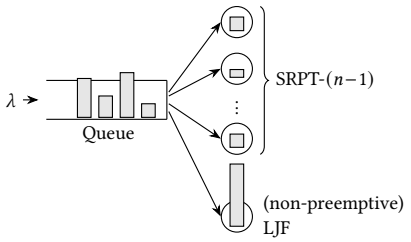


Fig. 1. An illustration of the SPLIT policy. Jobs arrive to a central queue. The LJF server always fetches the largest job; the remaining $n-1$ SRPT servers serve the jobs with the smallest remaining sizes.

4.2 Main Theorem and Proof Sketch

In this subsection, we provide the main theorem, Theorem 4.2, and a high-level proof outline (we provide the results needed to complete the proof in Section 4.3 and 4.4). The argument that SPLIT is strongly tail optimal is an immediate corollary of Theorem 4.2, via Lower Bound 2 in Section 3.2.

Theorem 4.2. *Suppose the load ρ satisfies $\rho < \frac{n-1}{n}$. Then the asymptotic response time tail under policy SPLIT is captured by the following equation:*

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{SPLIT} > t\right]}{\mathbb{P}\left[S > t\right]} = 1. \tag{7}$$

Proof. We prove the theorem by using a tagged job argument. Consider a tagged job with size x, and define its response time under policy SPLIT to be T_x^{SPLIT}. We now consider three cases for x (small, large, and extra large) and produce a different bound on $\mathbb{P}\left[T_x^{SPLIT} > t\right]$ in each case.

Case 1: Small jobs ($x \leq \frac{t}{C}$ for some constant $C > 0$). For these small jobs, we will show it is very unlikely that the job's response time exceeds t. In fact, we can prove that the probability that the job's response time exceeds t decays faster than any polynomial function of t. Mathematically, we prove the following inequality in Lemma 4.4: For any $\beta > 0$, there exists a constant $C > 0$ such that

$$\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, x \leq \frac{t}{C}\right] = o(t^{-\beta}). \tag{8}$$

Specifically, we pick a constant $\beta > \alpha$, where α is defined in (1), and take C to be the corresponding constant from Lemma 4.4.

Case 2: Large jobs ($\frac{t}{C} < x \leq t - \sqrt{t}$). For these large jobs, we show that they are very likely to end up being served by the (non-preemptive) LJF server before waiting for $\sqrt{x}$ time[1]. Mathematically, define $p(x)$ to be the probability that the tagged job gets served by the LJF server before waiting for $\sqrt{x}$ time. Then the probability that the tagged job has response time larger than t is bounded by $1 - p(x)$, i.e.,

$$\begin{aligned}
&\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, \frac{t}{C} < x \leq t - \sqrt{t}\right] \\
&\leq \mathbb{P}\left[T_x^{SPLIT} > x + \sqrt{x} \,\middle|\, \frac{t}{C} < x \leq t - \sqrt{t}\right] \\
&\leq 1 - \inf_{\frac{t}{C} < x \leq t - \sqrt{t}} p(x).
\end{aligned} \tag{9}$$

Lemma 4.11 proves that this probability, $p(x)$, converges to 1 as x goes to infinity, and hence the tail probability in (9) goes to 0.

Case 3: Extra large jobs ($x > t - \sqrt{t}$). For these extremely large jobs, we bound the probability that the tagged job has response time larger than t by 1, i.e.,

$$\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, x > t - \sqrt{t}\right] \leq 1. \tag{10}$$

Now we combine the bounds for the above three cases, dividing them by $\mathbb{P}[S > t]$ and taking the limit as t goes to infinity: First, we have that

$$\begin{aligned}
\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{SPLIT} > t\right]}{\mathbb{P}[S > t]} \leq \lim_{t\to\infty} \frac{1}{\mathbb{P}[S > t]} \Bigg(&\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, x \leq \frac{t}{C}\right] \mathbb{P}\left[S \leq \frac{t}{C}\right] \\
&+ \mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, \frac{t}{C} < x \leq t - \sqrt{t}\right] \mathbb{P}\left[\frac{t}{C} < S \leq t - \sqrt{t}\right] + \mathbb{P}\left[S > t - \sqrt{t}\right] \Bigg).
\end{aligned} \tag{11}$$

Then by (8) we have that the first term is

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, x \leq \frac{t}{C}\right] \mathbb{P}\left[S \leq \frac{t}{C}\right]}{\mathbb{P}[S > t]} \leq \lim_{t\to\infty} \frac{o(t^{-\beta})}{\mathbb{P}[S > t]} = 0.$$

[1]Here $\sqrt{x}$ is not crucial and can be any other function in $o(x)$.

By (9) and Lemma 4.11, we have that the second term is

$$\begin{aligned}
&\lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \mathbb{P}\left[T_x^{SPLIT} > t \;\middle|\; \frac{t}{C} < x \le t - \sqrt{t}\right] \mathbb{P}\left[\frac{t}{C} < S \le t - \sqrt{t}\right] \\
&\le \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \cdot \mathbb{P}\left[T_x^{SPLIT} > t \;\middle|\; \frac{t}{C} < x \le t - \sqrt{t}\right] \mathbb{P}\left[S > \frac{t}{C}\right] \\
&\le \lim_{t\to\infty} \frac{\mathbb{P}\left[S > \frac{t}{C}\right]}{\mathbb{P}\left[S > t\right]} \cdot \left(1 - \inf_{\frac{t}{C} < x \le t - \sqrt{t}} p(x)\right) \\
&= C^{\alpha} \cdot 0 = 0.
\end{aligned}$$

And the third term, by using the property of regularly varying distribution (1), is

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[S > t - \sqrt{t}\right]}{\mathbb{P}\left[S > t\right]} = 1.$$

Substituting all three terms above into (11) gives the proof.

□

Corollary 4.3. *SPLIT is strongly tail optimal if the load ρ satisfies $\rho < \frac{n-1}{n}$.*

Proof. For any policy π', as stated in (6), we have that $\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\pi'} > t\right]}{\mathbb{P}\left[S > t\right]} \ge 1$. Hence by Theorem 4.2, we have that

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{SPLIT} > t\right]}{\mathbb{P}\left[T^{\pi'} > t\right]} = \lim_{t\to\infty} \frac{\mathbb{P}\left[T^{SPLIT} > t\right]}{\mathbb{P}\left[S > t\right]} \cdot \frac{\mathbb{P}\left[S > t\right]}{\mathbb{P}\left[T^{\pi'} > t\right]} \le 1.$$

□

4.3 Proof for the bound for small jobs

In this subsection, our main goal is to prove Lemma 4.4 that is used in the proof of Theorem 4.2.

Lemma 4.4. *Suppose $\rho < \frac{n-1}{n}$. Then for any $\beta > 0$, there exists a constant $C > 0$ such that*

$$\mathbb{P}\left[T_x^{SPLIT} > t \;\middle|\; x \le \frac{t}{C}\right] = o(t^{-\beta}). \tag{12}$$

We start with some rough intuitions on Lemma 4.4. While Lemma 4.4 holds for any job with size $x \le \frac{t}{C}$, it implicitly says different things for different job sizes x: (1) For those relatively large jobs (with size $x \approx \frac{t}{C}$), the lemma says that their response time is very unlikely to be larger than a constant times x; (2) For smaller jobs (with size $x \ll \frac{t}{C}$), their response time is likely to be smaller than the response time of relatively large jobs. Property (1) is guaranteed by the $n-1$ SRPT servers under SPLIT– a similar property is proved in Lemma A.2. Property (2), in contrast, holds automatically under SRPT-n (where there is monotonicity of response time with respect to job size), but not for SPLIT by design. In proving Lemma 4.4, we overcome this difficulty by bounding the response time under SPLIT via relevant work, and then leveraging the monotonicity of relevant work with respect to job size.

To rigorously prove this lemma, we proceed through a sequence of reductions (shown in Figure 2), relating the response time of the tagged job under SPLIT first to its counterpart in a single super-server system with speed $n-1$ running SRPT, and then to a single-server system with speed 1 running SRPT (which we call SRPT-1), where standard tail bounds apply.

We first define the following two notations for work.

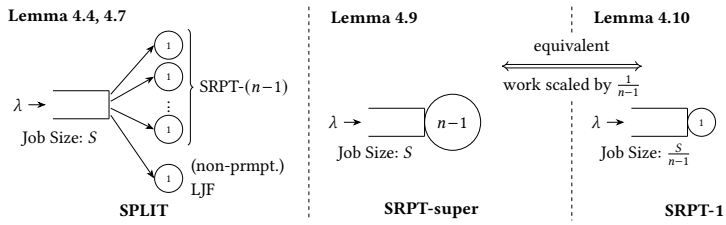


Fig. 2. Three systems used in the proof: the n-server system under SPLIT, a single super-server with speed $n-1$ under SRPT, and a single server with speed 1 under SRPT.

Definition 4.5 (Relevant work under SPLIT). Define the relevant work at time t, $W_x^{SPLIT}(t)$, to be the total work comprising jobs in the system at time t with remaining size smaller than x, excluding the job being served by the LJF server. Further define W_x^{SPLIT} to be the steady-state relevant work.

Definition 4.6 ($W_{A_x}(t)$). Define $W_{A_x}(t)$ to be the total amount of work comprising jobs arriving from time 0 to time t with size smaller than x.

We can now bound the response time of a job of size x by the above two types of work.

Lemma 4.7. *Without loss of generality, assume a tagged job of size x arrives at time 0. For this tagged job, we have that*

$$\mathbb{P}\left[T_x^{SPLIT} > t\right] \leq \mathbb{P}\left[x + \frac{1}{n-1}W_x^{SPLIT} + \frac{1}{n-1}W_{A_x}(t) > t\right]. \tag{13}$$

Proof. We will upper bound the work that needs to get done before the tagged job leaves the system. By PASTA (Poisson Arrivals See Time Averages, e.g., Section 13.3 in [21]), the relevant work that the tagged job sees upon arrival, not including the tagged job itself, has the distribution of the steady-state relevant work W_x^{SPLIT}. Intuitively speaking, this relevant work, together with newly arriving work comprising jobs with size smaller than x, must be completed with rate at least $(n-1)$. The only exception is when there are fewer than $(n-1)$ jobs with remaining size smaller than the tagged job, in which case the tagged job is served. The response time of the tagged job can be upper-bounded by the above argument.

To see this more precisely, we define a potential function $G(s)$ at time s to be the sum of three terms: (1) remaining size of the tagged job, (2) $\frac{1}{n-1}$ times the remaining work from the initial relevant work W_x^{SPLIT}, and (3) $\frac{1}{n-1}$ times the remaining work from the newly arrived work $W_A(x,s)$.

Hence, we have that $G(0) = x + \frac{1}{n-1}W_x^{SPLIT}$. The dynamics of $G(s)$ are given by: At arrivals, $G(s)$ increases by $\frac{1}{n-1}$ times the size of the arriving job if the arriving job is smaller than x. Between arrivals, $G(s)$ decreases with rate at least 1 as long as the tagged job is in the system (if the tagged job is not served, then all $n-1$ SRPT servers are serving jobs with size smaller than x, each reducing the potential function with rate $\frac{1}{n-1}$; otherwise if the tagged job is served, then the first term of the potential function already decreases with rate 1). Mathematically, integrating the decrease and adding the increases at arrivals, we have that for any s before the tagged job completes,

$$G(s) \leq x + \frac{1}{n-1}W_x^{SPLIT} + \frac{1}{n-1}W_{A_x}(s) - s. \tag{14}$$

Consider the event that the tagged job is not completed before time t (which means its response time is more than t). In this case, $G(t) > 0$ since the tagged job still has positive remaining size,

and (14) holds for $s = t$. Hence, we have that

$$0 < G(t) \leq x + \frac{1}{n-1} W_x^{SPLIT} + \frac{1}{n-1} W_{A_x}(t) - t. \tag{15}$$

Hence we have that

$$\mathbb{P}\left[T_x^{SPLIT} > t\right] \leq \mathbb{P}\left[x + \frac{1}{n-1} W_x^{SPLIT} + \frac{1}{n-1} W_{A_x}(t) > t\right].$$

□

Now we relate the relevant work under SPLIT to the relevant work under SRPT-super.

Definition 4.8 (Relevant work under SRPT-super). Define the relevant work in the SRPT-super system at time t, $W_x^{super}(t)$, to be the total work comprising jobs in the system at time t with remaining size smaller than x. Further define W_x^{super} to be the steady-state relevant work.

Lemma 4.9. *We have that under the same sample path, at any moment of time t,*

$$W_x^{SPLIT}(t) \leq W_x^{super}(t) + (n-1)x. \tag{16}$$

Proof. The proof is motivated by Lemma 2 (Section 5.2) in [17]. Consider the n-server system under policy SPLIT. Define a "few-jobs interval" to be a time interval where there are no more than $(n-1)$ jobs with remaining size smaller than x (excluding the job being served by the LJF server) in the n-server system under policy SPLIT. Define "many-jobs interval" to be the opposite. Then, for any time t within a few-jobs interval, we have $W_x^{SPLIT}(t) \leq (n-1) \cdot x$, and the lemma holds obviously.

For any time t within a many-jobs interval, we argue that the difference $W_x^{SPLIT}(t) - W_x^{super}(t)$ is non-increasing. This is because during many-jobs intervals, the $(n-1)$ SRPT servers in SPLIT work on the relevant work, and the relevant work decreases with rate at least $(n-1)$, which is no slower than the super server completes the relevant work in the super-server system. Throughout, the jobs served on the LJF server are excluded in both the relevant work and the determination of a "few-jobs interval", so only arrivals start a "many-jobs interval" or increase the relevant work. Moreover, the arrival sequences in both systems are the same, so arrivals do not affect the difference. □

We need one more lemma before we can prove Lemma 4.4. Lemma 4.10 is about a single server of speed 1 running SRPT (we refer to this as an SRPT-1 system), and slightly strengthens a lemma in [27]. The lemma in [27] says that, for any $\beta > 0$, there exists a constant C such that $\mathbb{P}\left[T_x^{SRPT-1} > Cx\right] = o(x^{-\beta})$.

Lemma 4.10 (SRPT-1 work bound). *For a single-server (speed 1) system under SRPT with load $\rho < 1$, we have that for any $\beta > 0$, there exists a constant $C > 0$ such that*

$$\mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(Cx) > Cx\right] = o(x^{-\beta}), \tag{17}$$

where W_x^{SRPT-1} refers to the (relevant) work in the SRPT-1 system consisting of jobs of remaining size $\leq x$ and $W_{A_x}^{SRPT-1}(Cx)$ refers to the total relevant arriving work into the SRPT-1 system by time Cx.

Proof. See Lemma C.1 in the Appendix. □

We are now ready to prove Lemma 4.4.

Proof of Lemma 4.4. Our goal is to bound T_x^{SPLIT} where $x \leq \frac{t}{C}$ is the size of a tagged job. We choose to let C be the constant from Lemma 4.10.

By Lemma 4.7,

$$\mathbb{P}\left[T_x^{SPLIT} > t\right] \leq \mathbb{P}\left[x + \frac{1}{n-1}W_x^{SPLIT} + \frac{1}{n-1}W_{A_x}(t) > t\right]. \tag{18}$$

By Lemma 4.9, $W_x^{SPLIT} \leq_{st} W_x^{super} + (n-1)x$. Moreover, by definition, $W_x^{SRPT-1} = \frac{1}{n-1}W_x^{super}$ and $W_{A_x}^{SRPT-1}(t) = \frac{1}{n-1}W_{A_x}(t)$. Hence we have that

$$\begin{aligned}
&\mathbb{P}\left[T_x^{SPLIT} > t \,\middle|\, x \leq \frac{t}{C}\right] \\
&\leq \mathbb{P}\left[x + \frac{1}{n-1}W_x^{SPLIT} + \frac{1}{n-1}W_{A_x}(t) > t \,\middle|\, x \leq \frac{t}{C}\right] && \text{by Lemma 4.7} \\
&\leq \mathbb{P}\left[x + \frac{1}{n-1}(W_x^{super} + (n-1)x) + \frac{1}{n-1}W_{A_x}(t) > t \,\middle|\, x \leq \frac{t}{C}\right] && \text{by Lemma 4.9} \\
&= \mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(t) > t \,\middle|\, x \leq \frac{t}{C}\right] \\
&\leq \mathbb{P}\left[2\left(\frac{t}{C}\right) + W_{\frac{t}{C}}^{SRPT-1} + W_{A_{\frac{t}{C}}}^{SRPT-1}(t) > t\right] \\
&= o\left(\left(\frac{t}{C}\right)^{-\beta}\right) = o(t^{-\beta}). && \text{by Lemma 4.10}
\end{aligned}$$

□

4.4 Proof for the bound for large jobs

In this section, we prove the following lemma that is used in the proof of Theorem 4.2. The lemma shows that for a sufficiently large job, it is very likely that under policy SPLIT, the job gets served by the (non-preemptive) LJF server before waiting for $\sqrt{x}$ time.

Lemma 4.11. *Under policy SPLIT, define $p(x)$ to be the probability that a job with size x gets served by the LJF server before waiting for $\sqrt{x}$ time. Then we have that*

$$\lim_{x\to\infty} p(x) = 1. \tag{19}$$

Proof. For the tagged job with size x, we consider the following two events:

(1) When the tagged job arrives, it sees no job with original size larger than $\sqrt{x}$. Define the probability of this event to be $p_1(x)$.
(2) The tagged job does not see any job with size larger than x arriving during the $\sqrt{x}$ time after it arrives. Define the probability of this event to be $p_2(x)$.

Note that the above two events are independent. Moreover, we know that if both of the above two events happen, the tagged job gets served by the LJF server immediately after the LJF server finishes the remaining work of the job being served (which must be smaller than $\sqrt{x}$ because event (1) happens). Hence we have that

$$p(x) \geq p_1(x)p_2(x). \tag{20}$$

Now we characterize $p_1(x)$ and $p_2(x)$ respectively.

Characterizing $p_1(x)$. Because the arrival process is Poisson, by the PASTA theorem (e.g., Section 13.3 in [21]), the probability that the tagged job sees no job with original size $> \sqrt{x}$ upon arrival is equal to the time average probability that the system does not have jobs with original size $> \sqrt{x}$.

We define "red jobs" to be the jobs with original size larger than $\sqrt{x}$. Define $W_{red}(t)$ to be the total remaining work in the system from red jobs at time t. Then by the above PASTA argument:

$$p_1(x) = \lim_{t\to\infty} \frac{\int_0^t \mathbf{1}[W_{red}(s) = 0]ds}{t}.$$

We know that under policy SPLIT, if there exists a red job in the system, either: (1) The LJF server is serving some red job; or (2) The LJF server will start serving red jobs after it finishes the remaining work of the job being served (if at that time red jobs still exist). Moreover, if the LJF server starts serving red jobs, it will keep serving red jobs until there are no red jobs in the system. Hence, suppose at time t a red job arrives and $W_{red}(t) > 0$ (while $W_{red}(t-) = 0$). Then either W_{red} immediately decreases with rate at least 1 between red arrivals until it drops to 0 (note that during this period, there could be also SRPT servers serving red jobs), or W_{red} starts decreasing with rate at least 1 after a time delay of at most $\sqrt{x}$ (because the LJF server is currently serving a job with original size at most $\sqrt{x}$). See Figure 3 for an illustration.

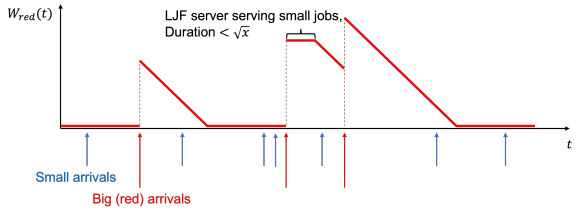


Fig. 3. The dynamic of $W_{red}(t)$ under policy SPLIT.

During each red work period (the consecutive period when $W_{red}(t) > 0$), we know that for at most $\sqrt{x}$ time, W_{red} is not decreasing; for the rest of the time, W_{red} is decreasing with rate at least 1 between red arrivals. Define B to be the duration of the red work period, and W to be the total red work arriving during this period including the red jobs that start the red work period (clearly $W > \sqrt{x}$). Then, since the total decrease of W_{red} during the period equals W,

$$B \le \sqrt{x} + W < 2W.$$

Define $B^{(i)}$ to be the duration of the i-th red work period, which ends at time $t^{(i)}$, and $W^{(i)}$ to be the total red work arriving during the i-th red work period (including the starting red jobs). Then

$$p_1(x) = \lim_{t\to\infty} \frac{\int_0^t \mathbf{1}[W_{red}(s) = 0]ds}{t} = 1 - \lim_{i\to\infty} \frac{\sum_{j=1}^{i} B^{(j)}}{t^{(i)}} \ge 1 - \lim_{i\to\infty} \frac{2\sum_{j=1}^{i} W^{(j)}}{t^{(i)}} = 1 - 2r_{>\sqrt{x}},$$

where $r_{>x}$ was defined in (2).

Characterizing $p_2(x)$. The arrival process of jobs larger than x is a Poisson process with rate $\lambda \cdot \mathbb{P}[S > x]$. Hence, the probability that no red job arrives during the $\sqrt{x}$ time after the tagged job arrives is given by

$$p_2(x) = e^{-\lambda\cdot\mathbb{P}[S>x]\sqrt{x}}.$$

Hence, we have that $p(x) \ge p_1(x)p_2(x) \ge (1 - 2r_{>\sqrt{x}})e^{-\lambda\cdot\mathbb{P}[S>x]\sqrt{x}}$, where both terms on the right-hand side converge to 1 as x goes to infinity. □

5 Thresh-SPLIT(d): tail optimal under all loads

In this section, we introduce the Thresh-SPLIT(d) policy (see Figure 4). We show that for load $\rho \geq \frac{n-1}{n}$, when the threshold parameter d is properly chosen, Thresh-SPLIT(d) can be made arbitrarily close to strongly tail optimal. For completeness, we show that Thresh-SPLIT(d) achieves this for load $\rho < \frac{n-1}{n}$ as well, although SPLIT is already strongly tail optimal in that regime. In contrast, Thresh-SPLIT(d) only approaches strong tail optimality as d approaches the limiting value in Theorem 5.4; for any fixed d, it is not strongly tail optimal.

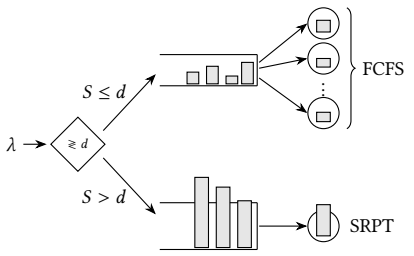


Fig. 4. An illustration of the Thresh-SPLIT(d) policy. Jobs are classified by size: small jobs ($S \leq d$) are served by $n-1$ FCFS servers, and big jobs ($S > d$) are served by a single SRPT server.

Definition 5.1 (Thresh-SPLIT(d)). Parameterized by a threshold d, the policy Thresh-SPLIT(d) classifies all jobs as either *big*, if their size is $> d$, or *small*, if their size is $\leq d$. It then splits the n servers into $n-1$ servers reserved for small jobs and 1 server reserved for big jobs. The $n-1$ small-job servers apply FCFS (with a central queue) and the 1 big-job server applies policy SRPT.

The next lemma is a conventional result for the response time tail of M/G/n systems with light-tailed job sizes under FCFS.

Lemma 5.2 (Theorem 2 in [30]). *For any FCFS M/G/n system with light-tailed job sizes and load $\rho < 1$, let T denote the response time in this system. Then there exists a constant $\gamma > 0$ such that*

$$\lim_{t\to\infty} \frac{\mathbb{P}[T > t]}{e^{-\gamma t}} \leq 1.$$

Lemma 5.2 is precisely what we need to handle the small-job side of Thresh-SPLIT(d). The $n-1$ small-job servers only serve jobs with size bounded by d, so their size distribution is trivially light-tailed. Thus by Lemma 5.2 the response time tail of the small jobs decays exponentially. Since the overall response time tail is heavy-tailed (driven by the big jobs), this exponential decay implies that small jobs do not contribute to the asymptotic response time tail of the system.

Moreover, since small jobs' sizes are bounded by d, the same exponentially decaying response time tail should hold under most reasonable scheduling policies on the $n-1$ small-job servers, e.g., SRPT-$(n-1)$. We choose FCFS in Definition 5.1 because Lemma 5.2 gives a rigorous proof in that case, whereas to our knowledge no analogous formal result is available for multi-server SRPT with bounded sizes. We revisit this design choice in Section 8.1 and validate it empirically in Section 7 and Appendix E.1.

Under policy Thresh-SPLIT(d), the load on the single big-job server is $r_{>d}$ (recall (2)). The resource requirement on the $n-1$ small-job servers is then

$$r_{\leq d} := \rho \cdot n - r_{>d}.$$

For the small-job system to be stable, we need $r_{\leq d} < n-1$, i.e., the load on the small system must be < 1. Given this stability condition, we prove Theorem 5.3, from which we will derive our main theorem, Theorem 5.4, on the tail optimality of Thresh-SPLIT(d).

Theorem 5.3. *Under policy Thresh-SPLIT(d), if $r_{\leq d} < n-1$ (the small-job system is stable), then*

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\textit{Thresh-SPLIT}(d)} > t\right]}{\mathbb{P}[S > (1-r_{>d})t]} = 1. \tag{21}$$

Proof. **Small jobs.** Since $r_{\leq d} < n-1$, the small-job system (the $n-1$ servers serving only jobs with size $\leq d$) has load smaller than 1. By Lemma 5.2, there exists a constant $\gamma > 0$ such that

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t \mid S \leq d\right]}{e^{-\gamma t}} \leq 1. \tag{22}$$

Big jobs. On the single big-job server, SRPT-1 achieves the asymptotic tail given by (3). Hence,

$$1 = \lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t \mid S > d\right]}{\mathbb{P}\left[S > (1-r_{>d})t \mid S > d\right]} \tag{23}$$

$$= \lim_{t\to\infty} \mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t \mid S > d\right] \cdot \frac{\mathbb{P}\left[S > d\right]}{\mathbb{P}\left[S > (1-r_{>d})t\right]}. \tag{24}$$

Combining. By the law of total probability,

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t\right]}{\mathbb{P}\left[S > (1-r_{>d})t\right]}$$
$$= \lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t \mid S \leq d\right]\mathbb{P}\left[S \leq d\right] + \mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t \mid S > d\right]\mathbb{P}\left[S > d\right]}{\mathbb{P}\left[S > (1-r_{>d})t\right]}.$$

The first term vanishes because the exponential decay in (22) is faster than any heavy-tailed distribution. The second term converges to 1 by (24). □

Theorem 5.4. *By choosing the threshold d appropriately, Thresh-SPLIT(d) can be made arbitrarily close to strongly tail optimal. Specifically, the following asymptotic tails match the lower bounds in Section 3.2:*

- *If $\rho \leq \frac{n-1}{n}$, we have that*

$$\lim_{d\to\infty}\left(\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\textit{Thresh-SPLIT}(d)} > t\right]}{\mathbb{P}\left[S > t\right]}\right) = 1. \tag{25}$$

- *If $\rho > \frac{n-1}{n}$, we have that*

$$\lim_{d\to(d^*)^-}\left(\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\textit{Thresh-SPLIT}(d)} > t\right]}{\mathbb{P}\left[S > n(1-\rho)t\right]}\right) = 1, \tag{26}$$

  where d^* is the smallest threshold such that $r_{\leq d^*} = (n-1)$.

Proof. Both cases follow from Theorem 5.3.

If $\rho \leq \frac{n-1}{n}$, then for any $d > 0$ the small-job system has load less than 1. As $d \to \infty$, we have $r_{>d} \to 0$, so by Theorem 5.3, the tail ratio converges to:

$$\lim_{d\to\infty}\left(\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t\right]}{\mathbb{P}\left[S > t\right]}\right) = \lim_{d\to\infty}\lim_{t\to\infty} \frac{\mathbb{P}\left[S > (1-r_{>d})t\right]}{\mathbb{P}\left[S > t\right]} = \lim_{d\to\infty}(1-r_{>d})^{-\alpha} = 1.$$

If $\rho > \frac{n-1}{n}$, stability of the small-job system requires $r_{>d} = n\rho - r_{\leq d} > n\rho - (n-1)$. As d increases toward d^* from below, $r_{>d}$ decreases toward $n\rho-(n-1)$, and the factor $(1-r_{>d})$ in (21) approaches $n(1-\rho)$. Substituting into Theorem 5.3 gives (26). □

6 TAGS-SPLIT(d)

In this section, we introduce the TAGS-SPLIT(d) policy, a variant of Thresh-SPLIT(d) that achieves tail optimality even when the scheduler does not know exact job sizes upon arrival. Inspired by the Task-Assignment-by-Guessing-Size (TAGS) policy [20], TAGS-SPLIT(d) identifies big jobs by running all jobs for a fixed time d and checking whether they complete.

Definition 6.1 (TAGS-SPLIT(d)). The policy TAGS-SPLIT(d), parameterized by a threshold d, splits the n servers into $n-1$ small-job servers operating under FCFS and 1 big-job server running Processor Sharing (PS), as shown in Figure 5. Every arriving job initially enters the FCFS system as a small job. If a job has been served for d time units without completing, it is identified as a big job and migrated to the big-job server in a preempt-resume manner (no work is lost). The PS server schedules all big jobs by giving every job an equal share of the server at every moment of time.

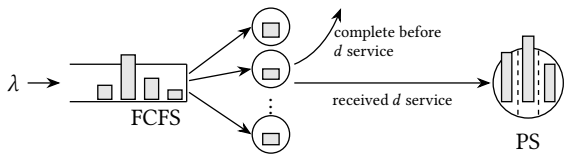


Fig. 5. An illustration of the TAGS-SPLIT(d) policy. All jobs initially enter the FCFS system. Jobs that have received d service without completing are migrated to the PS server.

TAGS-inspired policies rely on running jobs in upstream servers until a job has received a certain amount of service without completing; such a job is then deemed large and migrated to a downstream server. A central difficulty in the analysis of TAGS-inspired policies is that, even when the external arrivals to the upstream servers are Poisson, the arrival process to the downstream servers is neither Poisson nor renewal in general, since it is the upstream departure process thinned to jobs whose size exceeds the threshold. Most prior work in the TAGS literature has handled this difficulty by approximating the downstream arrival process as Poisson [4, 5, 20]. Empirical experiments have been used to support the accuracy of this approximation [4, 20]. We make the analogous approximation here.

Approximation 1 (Poisson arrivals at the big-job server). *In the analysis of TAGS-SPLIT(d), we assume that the arrival process to the PS (big-job) server is a Poisson process with rate $\lambda \cdot \mathbb{P}[S > d]$.*

Approximation 1 preserves the long-run arrival rate $\lambda \mathbb{P}[S > d]$ of big jobs, but it ignores the dependence among migration times created by the FCFS queue at the small-job servers. We do not have a proof that this dependence does not affect the asymptotic tail; our simulations in Section 7.3, which do not use the approximation, agree with Theorem 6.3. Given Approximation 1, we can now apply the well-known response time tail of the M/G/1 queue under PS [19].

For a threshold d, define the load of the big-job server under TAGS-SPLIT(d) to be $\rho_{large}^{TAGS}(d)$:

$$\rho_{large}^{TAGS}(d) = \lambda \cdot \mathbb{P}[S > d] \cdot (\mathbf{E}[S \mid S > d] - d). \tag{27}$$

We now state Theorem 6.2 for TAGS-SPLIT(d), which is analogous to Theorem 5.3 for Thresh-SPLIT(d):

Theorem 6.2. *Under policy TAGS-SPLIT(d) and Approximation 1, if* $r_{small} := \rho \cdot n - \rho_{large}^{TAGS}(d) < n-1$ *(i.e., the small-job system is stable), then*

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{TAGS\text{-}SPLIT(d)} > t\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t\right]} = 1. \tag{28}$$

The proof follows the same structure as that of Theorem 5.3, splitting into small jobs (light-tailed) and big jobs (whose tail is governed by the PS server), and is deferred to Appendix D.

Given Theorem 6.2, we can now state the analogue of Theorem 5.4 for TAGS-SPLIT(d):

Theorem 6.3. *By choosing the threshold d appropriately, TAGS-SPLIT(d) can be made arbitrarily close to strongly tail optimal under Approximation 1. Specifically, the following asymptotic tails match the lower bounds in Section 3.2:*

- *If* $\rho \le \frac{n-1}{n}$, *we have that*

$$\lim_{d\to\infty}\left(\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{TAGS\text{-}SPLIT(d)} > t\right]}{\mathbb{P}\left[S > t\right]}\right) = 1. \tag{29}$$

- *If* $\rho > \frac{n-1}{n}$, *we have that*

$$\lim_{d\to(d^*)^-}\left(\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{TAGS\text{-}SPLIT(d)} > t\right]}{\mathbb{P}\left[S > n(1-\rho)t\right]}\right) = 1, \tag{30}$$

  where d^* *is the smallest threshold such that* $\rho \cdot n - \rho_{large}^{TAGS}(d^*) = n-1$.

Proof. The proof follows the same argument as Theorem 5.4, with $r_{>d}$ replaced by $\rho_{large}^{TAGS}(d)$. □

7 Simulation

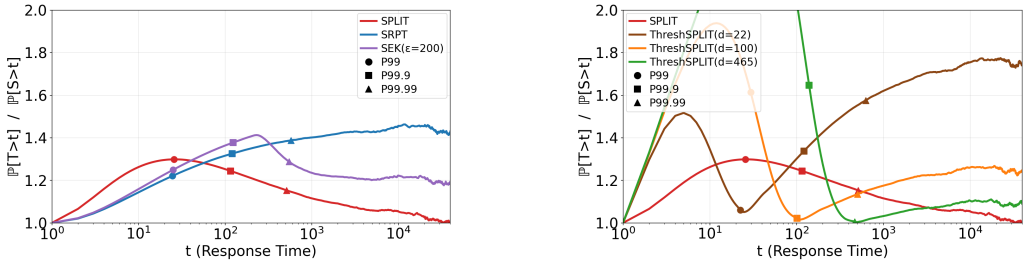
In this section we empirically evaluate our policies and compare them with several baselines. Each simulation runs an M/G/n system with Poisson arrivals, Pareto-distributed job sizes with $\mathbb{P}[S > t] = t^{-\alpha}$ for $t \ge 1$, and 10^{10} job arrivals.

Metric. Motivated by the lower bounds in Section 3.2, we plot the *normalized tail*, i.e., the response time tail $\mathbb{P}[T > t]$ divided by the applicable lower bound: for $\rho < \frac{n-1}{n}$, we plot $\mathbb{P}[T > t] / \mathbb{P}[S > t]$, and for $\rho \ge \frac{n-1}{n}$, we plot $\mathbb{P}[T > t] / \mathbb{P}[S > n(1-\rho)t]$. A normalized tail converging to 1 as $t \to \infty$ corresponds to strong tail optimality.

Baselines. We compare against SRPT-n, and SEK-ϵ; here SEK-ϵ refers to the Practical-SEK policy of [16], which mostly follows SRPT-n except when the system contains exactly $n+1$ jobs with one of size larger than ϵ and the other n of size smaller than ϵ, in which case it serves the largest job in place of the second-largest. For SEK-ϵ, the asymptotic tail empirically improves as ϵ grows, but stops improving at $\epsilon = 200$. Hence, we plot only the SEK-ϵ curve with $\epsilon = 200$, which is best among the SEK family in our experiments. We provide SEK-ϵ curves for a range of ϵ in Appendix E.2. We do not include Boost [38, 39], which is only defined for light-tailed job size distributions.

Our policies. We simulate SPLIT, which applies only when $\rho < \frac{n-1}{n}$, and Thresh-SPLIT(d) for several values of d. In our experiments, Thresh-SPLIT(d) is implemented with a work-stealing improvement: when the large-job server is idle, it opportunistically serves small jobs, and the moment a job with size $> d$ arrives it preempts the job in service and returns to serving large jobs.

7.1 Low load ($\rho < \frac{n-1}{n}$)



(a) SPLIT and baselines

(b) Thresh-SPLIT(d) for several thresholds d

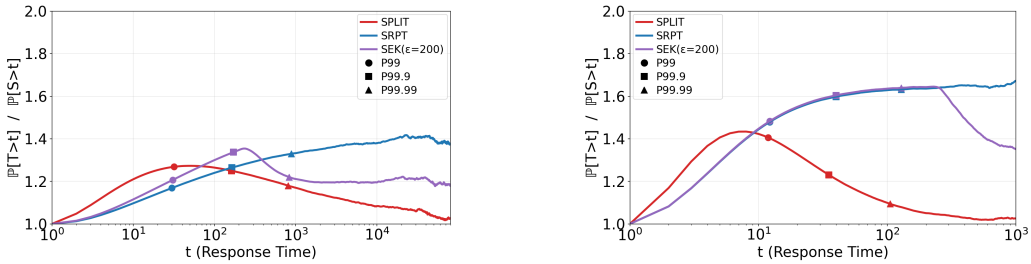
Fig. 6. Normalized response time tail, $n = 3$ servers, $\rho = 0.5$, Pareto job sizes with $\alpha = 1.5$.

Figure 6 shows results for $n = 3$, $\rho = 0.5$, and Pareto job sizes with tail index $\alpha = 1.5$ and mean $\frac{\alpha}{\alpha-1} = 3$. Figure 6a compares SPLIT (in red) with the baselines: SPLIT's normalized tail converges to 1, confirming its strong tail optimality (Theorem 4.2), whereas all baselines remain bounded away from 1. In particular, while we were able to prove that SRPT-n is weakly optimal (see Appendix A), Figure 6a confirms, as we prove in Appendix B, that SRPT-n is not strongly optimal.

SPLIT outperforms SRPT-n at the 99.9%-tile and 99.99%-tile of response time, but trails SRPT-n at the 99%-tile of response time in this setting. The loss in the 99%-tile of response time is not universal: in the $\alpha = 2.0$ experiment below, SPLIT also wins at the 99%-tile of response time (see Figure 7b).

Figure 6b reports Thresh-SPLIT(d) for three choices of the threshold d corresponding to the 99%, 99.9%, and 99.99%-tile of the job size distribution, respectively. Consistent with Theorem 5.4, the normalized tail approaches 1 as d grows.

Figure 7 varies the Pareto tail index α, fixing $n = 3$ and $\rho = 0.5$.



(a) $\alpha = 1.4$

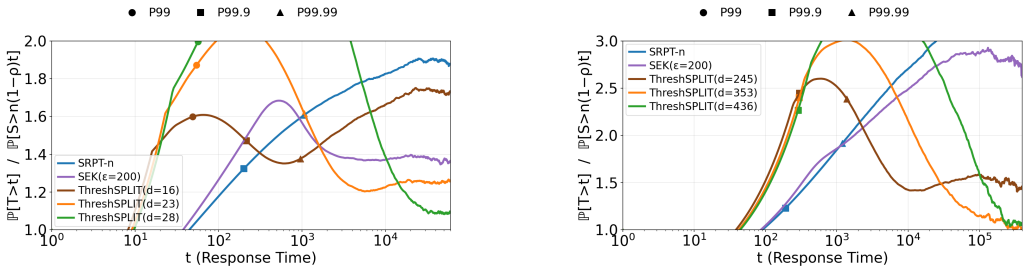
(b) $\alpha = 2.0$

Fig. 7. Normalized response time tail for varying Pareto tail index, $n = 3$ servers, $\rho = 0.5$.

The qualitative observations are preserved across both values of α: SPLIT's normalized tail converges to 1, the baselines remain bounded away from 1. In addition, at $\alpha = 2.0$, SPLIT attains better 99%, 99.9%, and 99.99%-tile of response times than SRPT-n. We do not have a clean explanation for this finite-percentile gain and report it without one.

7.2 High load ($\rho \geq \frac{n-1}{n}$)

Figure 8 shows two high-load settings: $n = 3$ with $\rho = 0.8$, and $n = 10$ with $\rho = 0.94$; both use Pareto job sizes with $\alpha = 1.5$. In both cases, keeping the small-job system stable requires the large-job server to carry more than 0.4 of the service requirement. We test thresholds corresponding to large-job-server load 0.45, 0.5, and 0.6. SPLIT does not apply in this regime. We observe that with properly chosen thresholds, Thresh-SPLIT(d) approaches strong tail optimality and outperforms all baselines asymptotically.

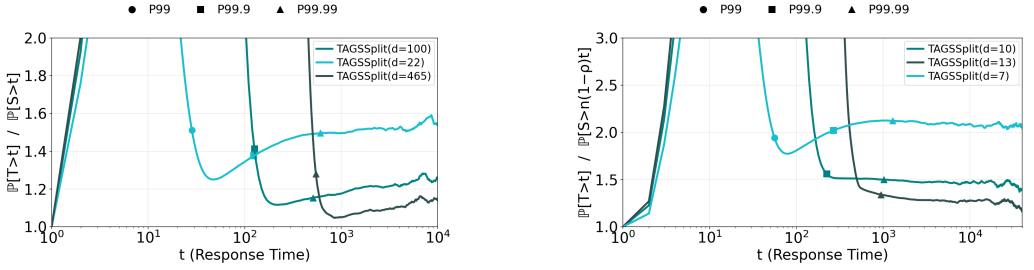


(a) $n = 3$, $\rho = 0.8$ (b) $n = 10$, $\rho = 0.94$

Fig. 8. Normalized response time tail in the high-load regime, Pareto job sizes with $\alpha = 1.5$. Thresh-SPLIT(d) uses SRPT-$(n-1)$ on the small-job servers.

In this regime, we use SRPT-$(n-1)$ on the small-job servers of Thresh-SPLIT(d) rather than FCFS. As discussed in Section 5, this choice is unlikely to affect the asymptotic tail: Conceivably both FCFS and SRPT-$(n-1)$ yield exponentially decaying response time tails for the bounded-size small jobs. Empirically, however, FCFS produces a noticeably larger bump in the pre-asymptotic regime at high load, which motivates the SRPT-$(n-1)$ choice here; a direct side-by-side comparison is given in Appendix E.1.

7.3 Performance of TAGS-SPLIT(d)



(a) Low load: $n = 3$, $\rho = 0.5$ (b) High load: $n = 3$, $\rho = 0.8$

Fig. 9. Normalized response time tail of TAGS-SPLIT(d), Pareto job sizes with $\alpha = 1.5$.

Figure 9 evaluates TAGS-SPLIT(d) in the same two settings as Sections 7.1 and 7.2. The results match Theorem 6.3 in both regimes. Figure 9a uses the same three thresholds as Thresh-SPLIT(d) in Figure 6b. As d grows, the normalized tail of TAGS-SPLIT(d) approaches 1, consistent with

Theorem 6.3. Figure 9b shows the high-load setting, with thresholds chosen so that the big-job-server load $\rho_{large}^{TAGS}(d)$ is 0.45, 0.5, and 0.6 (recall the definition in (27)). The normalized tail again approaches 1 as $\rho_{large}^{TAGS}(d)$ decreases toward the critical value $\rho n - (n-1) = 0.4$.

8 Discussion

The policies we propose in this paper, SPLIT, Thresh-SPLIT(d), and TAGS-SPLIT(d), are all guided by two design intuitions. First, because it is the large jobs that dominate the response time tail, we dedicate separate service capacity to large jobs to reduce their response time. Second, small jobs should not be blocked by large jobs; in particular, a small job's response time should not be inflated by a catastrophically large job in the system. In this section, we reexamine these two intuitions from two perspectives: policy design (Section 8.1) and the effect of packing (Section 8.2).

8.1 Policy design

SPLIT and Thresh-SPLIT(d) share a common structure: The n servers are split into two groups, one server dedicated to large jobs and the remaining $n-1$ servers dedicated to small jobs. Despite the shared structure, SPLIT and Thresh-SPLIT(d) schedule jobs within each group very differently: SPLIT uses non-preemptive LJF on the large-job server and SRPT-$(n-1)$ on the small-job servers, while Thresh-SPLIT(d) uses SRPT-1 on the large-job server and FCFS on the small-job servers. We now explain the reason behind each of these choices, and distinguish the essential parts from those chosen for ease of analysis.

Scheduling the small jobs. Under SPLIT, the small-job servers must use a size-aware policy such as SRPT-$(n-1)$; FCFS would not suffice. The reason is that the LJF server handles only the single largest job in the system at a time, so whenever several big jobs are present, all but the largest remain at the small-job servers. SRPT-$(n-1)$ automatically deprioritizes these extra big jobs by their large remaining size, so small jobs continue to be served. FCFS, by contrast, would let those big jobs sit at the head of the small-job queue and potentially block subsequent small job arrivals.

Under Thresh-SPLIT(d), this issue does not arise: Every job with size $> d$ is routed to the large-job server, so the small-job servers never see a big job. Since the small jobs' sizes are bounded by d, the response time tail from the small-job servers should decay exponentially under any reasonable policy, such as FCFS or SRPT-$(n-1)$ (Lemma 5.2); our choice of FCFS is only for ease of analysis.

Scheduling the large jobs. The same design intuition, that a job should not be blocked by catastrophically larger jobs, now applies one level up: A merely large job should not be blocked by a catastrophically large job either. Under SPLIT, blocking between large jobs never occurs, because the LJF server holds only one job at a time and any merely large jobs remain at the small-job servers, where SRPT-$(n-1)$ can eventually serve them. This is why SPLIT can afford LJF on the large-job server; the non-preemptive part is adopted only for ease of analysis.

Under Thresh-SPLIT(d), by contrast, every job of size $> d$ is routed to a single large-job server, so the scheduling policy there must now prevent blocking between large jobs. If Thresh-SPLIT(d) used LJF on the large-job server, every merely large job would be blocked by any catastrophically large job present, inflating its response time. Scheduling the large jobs according to SRPT avoids this. More generally, any single-server policy that is strongly tail optimal under heavy tails, such as PS, FB, or SMART, would work equally well on the large-job server.

8.2 Tail benefit of SPLIT from improved packing

It is natural to ask why the SPLIT algorithm should not also be applied within a single-server system: One could give large jobs some partial priority in a single-server queue, without delaying

small jobs too much. In this subsection, we show that the tail benefit of SPLIT (compared to SRPT-n) intrinsically arises from its ability to pack work across servers better. Since packing is inherently a multi-server phenomenon, no analogous benefit can be obtained in a single-server system. We develop this view in the two-server case ($n = 2$) with $\rho < 1/2$ for clarity of exposition; the same intuition extends to $\rho \geq 1/2$ via Thresh-SPLIT(d), as we discuss at the end.

The catastrophe principle. Under heavy-tailed job sizes, extreme response times are driven by the *catastrophe principle*: A very large response time is typically caused by a single very large job in the system, rather than by a conspiracy of many moderately large jobs [25]. Understanding how a policy shapes the asymptotic response time tail therefore reduces to understanding how the policy handles a single catastrophically large job. Concretely, consider the typical scenario that produces a large response time: The system contains one very large job of size S^*; all other jobs currently in the system are much smaller, as are the jobs arriving during the lifetime of this large job. A good policy must simultaneously (i) keep small jobs' response times small, so that they do not themselves contribute to the tail, and (ii) make the large job's response time, T^*, as small as possible.

Packing and the response time of the large job. We quantify how well a policy *packs* by P_{idle}, the fraction of the large job's service time during which the other server is idle. The total idle server-time over the large job's lifetime is therefore $P_{\text{idle}} \cdot S^*$. Over an interval of length T^* after the arrival of the large job, the total work arriving to the system is approximately $2\rho\, T^*$, all of which must complete by time T^* under goal (i). Combined with the S^* units of work from the large job itself and the idle server-time $P_{\text{idle}} \cdot S^*$, work conservation gives $2\,T^* \approx S^* + P_{\text{idle}} \cdot S^* + 2\rho\, T^*$, which yields

$$T^* \approx \frac{(1 + P_{\text{idle}})\, S^*}{2(1-\rho)}.$$

Hence, to reduce the asymptotic tail, one wants P_{idle} as small as possible.

Comparing SRPT-2 and SPLIT. Under SRPT-2, the small jobs behave approximately as fluid: as long as at least two small jobs are present, which is almost always the case, the large job is preempted. The large job therefore runs essentially only when it is alone in the system, giving $P_{\text{idle}} \approx 1$, the worst possible packing.

Under SPLIT, by contrast, the large job is routed to the LJF server and runs there continuously throughout its lifetime. During this time, the other server handles the stream of small jobs as a single-server queue with load 2ρ; thus busy 2ρ fraction of time. Hence $P_{\text{idle}} \approx 1 - 2\rho$, the best packing any work-conserving policy can achieve, which is why SPLIT attains strong tail optimality.

Extension to $\rho \geq 1/2$. The same formula for T^* continues to apply when $\rho \geq 1/2$, and in this regime the goal is to drive P_{idle} close to 0. This is precisely what Thresh-SPLIT(d) accomplishes by deliberately choosing d such that the load on the small-job server is ≈ 1, keeping it busy almost all of the time during the lifetime of the catastrophically large job.

9 Conclusion

This paper develops the first strongly tail-optimal scheduling policies for the M/G/n queue with heavy-tailed job sizes. Our central finding is that tail optimality in multi-server systems calls for a qualitatively different principle than in single-server systems: Instead of strictly prioritizing short jobs, one should give large jobs a small amount of "sympathy" by dedicating a server to them. This sympathy improves the asymptotic tail by enabling the system to pack work across servers more efficiently, a benefit that has no single-server analog (see Section 8.2). Our three policies SPLIT, Thresh-SPLIT(d), and TAGS-SPLIT(d) realize this principle across the full stability region, with and without knowledge of job sizes.

Several natural directions remain open. The primary open problem is whether exact strong tail optimality can be achieve under high load (i.e., $\rho \geq \frac{n-1}{n}$). Recall that while Thresh-SPLIT(d) can be arbitrarily close to strongly tail optimal, it never achieves exact strong tail optimality. Second, tail-optimal scheduling for multi-server systems with *light-tailed* job sizes is only partially understood: Boost [38] is strongly tail optimal in heavy traffic, but nothing is known outside of heavy traffic. Third, it is interesting to ask whether a policy can be designed that optimizes both the mean and tail of response time.

Acknowledgments

This work was supported by NSF-CIF-2403194, NSF-III-2322973, and NSF-CMMI-2307008.

References

[1] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. 2009. On the Gittins Index in the M/G/1 Queue. *Queueing Systems* 63 (2009), 437–458.
[2] Joseph Abate, Gagan L. Choudhury, and Ward Whitt. 1995. Exponential Approximations for Tail Probabilities in Queues, I: Waiting Times. *Operations Research* 43, 5 (1995), 885–901.
[3] Venkat Anantharam. 1988. How large delays build up in a GI/G/1 queue. *Queueing Systems* 5, 4 (1988), 345–368.
[4] Eitan Bachmat, Josu Doncel, and Hagit Sarfati. 2019. Performance and Stability Analysis of the Task Assignment Based on Guessing Size Routing Policy. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 1–13.
[5] Eitan Bachmat, Josu Doncel, and Hagit Sarfati. 2020. Analysis of the Task Assignment based on Guessing Size policy. *Performance Evaluation* 142 (2020), 102122.
[6] Jose Blanchet and Karthyek Murthy. 2016. Tail Asymptotics for Delay in a Half-Loaded GI/GI/2 Queue with Heavy-Tailed Job Sizes. *Queueing Systems* 81, 4 (2016), 301–340.
[7] Sem Borst, Onno J. Boxma, Rudesindo Núñez-Queija, and Bert Zwart. 2003. The Impact of the Service Discipline on Delay Asymptotics. *Performance Evaluation* 54 (2003), 175–206.
[8] Sem Borst, Rudesindo Núñez-Queija, and Bert Zwart. 2006. Sojourn time asymptotics in processor-sharing queues. *Queueing Systems* 53, 1 (2006), 31–51.
[9] Onno J. Boxma, Qing Deng, and Bert Zwart. 2002. Waiting-Time Asymptotics for the M/G/2 Queue with Heterogeneous Servers. *Queueing Systems* 40 (2002), 5–31.
[10] Onno J. Boxma and Denis Denisov. 2011. Sojourn Time Tails in the Single Server Queue with Heavy-Tailed Service Times. *Queueing Systems* 69 (2011), 101–119.
[11] Onno J. Boxma and Bert Zwart. 2007. Tails in scheduling. *ACM SIGMETRICS Performance Evaluation Review* 34, 4 (2007), 13–20.
[12] Nils Charlet and Benny Van Houdt. 2024. Tail Optimality and Performance Analysis of the Nudge-M Scheduling Algorithm. *arXiv preprint arXiv:2403.06588* (2024).
[13] Mark E. Crovella and Azer Bestavros. 1997. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking* 5, 6 (1997), 835–846.
[14] Serguei Foss and Dmitry Korshunov. 2006. Heavy tails in multi-server queue. *Queueing Systems* 52, 1 (2006), 31–48.
[15] Serguei Foss and Dmitry Korshunov. 2012. On Large Delays in Multi-Server Queues with Heavy Tails. *Mathematics of Operations Research* 37, 2 (2012), 201–218.
[16] Izzy Grosof and Daniela Hurtado-Lange. 2025. Outperforming Multiserver SRPT at All Loads. *arXiv preprint arXiv:2510.25963* (2025).
[17] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2018. SRPT for multiserver systems. *Performance Evaluation* 127-128 (2018), 154–175.
[18] Isaac Grosof, Kunhe Yang, Ziv Scully, and Mor Harchol-Balter. 2021. Nudge: Stochastically Improving upon FCFS. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 2 (2021), 1–29.
[19] Fabrice Guillemin, Philippe Robert, and Bert Zwart. 2004. Tail asymptotics for processor-sharing queues. *Advances in Applied Probability* 36, 2 (2004), 525–543.
[20] M. Harchol-Balter. 2000. Task assignment with unknown duration. In *Proceedings 20th IEEE International Conference on Distributed Computing Systems*. 214–224.
[21] Mor Harchol-Balter. 2013. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press.
[22] Mor Harchol-Balter and Allen B. Downey. 1997. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems* 15, 3 (1997), 253–285.

[23] Amit Harlev, George Yu, and Ziv Scully. 2025. A Gittins Policy for Optimizing Tail Latency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 9, 2 (2025), 1–40.
[24] Robert M Loynes. 1962. The stability of a queue with non-independent inter-arrival and service times. In *Mathematical proceedings of the cambridge philosophical society*, Vol. 58. Cambridge University Press, 497–520.
[25] Jayakrishnan Nair, Adam Wierman, and Bert Zwart. 2022. *The Fundamentals of Heavy Tails: Properties, Emergence, and Estimation.* Cambridge University Press.
[26] Rudesindo Núñez-Queija. 2002. Queues with Equally Heavy Sojourn Time and Service Requirement Distributions. *Annals of Operations Research* 113 (2002), 101–117.
[27] Misja Nuyens, Adam Wierman, and Bert Zwart. 2008. Preventing large sojourn times using SMART scheduling. *Operations Research* 56, 1 (2008), 88–101.
[28] Misja Nuyens and Bert Zwart. 2006. A Large-Deviations Analysis of the GI/GI/1 SRPT Queue. *Queueing Systems* 54 (2006), 85–97.
[29] Sidney Resnick and Gennady Samorodnitsky. 1999. Activity periods of an infinite server queue and performance of certain heavy tailed fluid queues. *Queueing Systems* 33, 1 (1999), 43–71.
[30] John S. Sadowsky and Wojciech Szpankowski. 1990. On the Analysis of the Tail Queue Length and Waiting Time Distributions of a GI/G/c Queue. In *Proceedings of the 14th IFIP WG 7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation (Performance '90)*. North-Holland Publishing Co., NLD, 93–107.
[31] Linus Schrage. 1968. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. *Operations Research* 16, 3 (1968), 687–690.
[32] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2020. Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic. *Performance Evaluation* 145 (2020), 102150.
[33] Ziv Scully and Mor Harchol-Balter. 2021. The Gittins Policy in the M/G/1 Queue. In *19th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 248–255.
[34] Ziv Scully and Lucas van Kreveld. 2025. When Does the Gittins Policy Have Asymptotically Optimal Response Time Tail in the M/G/1? *Operations Research* 73, 3 (2025), 1412–1429.
[35] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing He Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–14.
[36] Benny Van Houdt. 2022. On the Stochastic and Asymptotic Improvement of First-Come First-Served and Nudge Scheduling. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 3 (2022), 1–22.
[37] Adam Wierman and Bert Zwart. 2012. Is tail-optimal scheduling possible? *Operations research* 60, 5 (2012), 1249–1257.
[38] George Yu, Amit Harlev, Reevu Adakroy, and Ziv Scully. 2025. A Tale of Two Traffics: Optimizing Tail Latency in the Light-Tailed M/G/k. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 9, 3 (2025), 1–40.
[39] George Yu and Ziv Scully. 2024. Strongly tail-optimal scheduling in the light-tailed M/G/1. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 2 (2024), 1–33.
[40] Bert Zwart and Onno J. Boxma. 2000. Sojourn Time Asymptotics in the M/G/1 Processor Sharing Queue. *Queueing Systems* 35 (2000), 141–166.

A SRPT-n is weakly tail optimal

In this section, we prove that SRPT-n is weakly tail optimal. The proof of the main lemma (Lemma A.2) is very similar to the proof in Section 4.3. The proof of the main theorem (Theorem A.1) follows a similar structure in Section 4.2. We state the whole proof for completeness.

Our main goal is to prove Theorem A.1 below:

Theorem A.1. *SRPT-n is weakly tail optimal.*

Define the response time of a job with size x in an n-server system under SRPT-n to be $T_x^{SRPT\text{-}n}$. The main lemma is stated below:

Lemma A.2. *For an n-server system under SRPT-n with Poisson arrivals and total load $\rho < 1$, for any $\beta > 0$, there exists a constant $C > 0$ such that*

$$\mathbb{P}\left[T_x^{SRPT\text{-}n} > Cx\right] = o(x^{-\beta}), \quad x \to \infty. \tag{31}$$

The proof of Lemma A.2 proceeds in three steps, mirroring the proof structure in Section 4.3. First, we use a potential function argument to bound the response time of a job under SRPT-n in

terms of the relevant work it sees upon arrival and the new small-job work that arrives during its response time (Lemma A.5). Second, we relate the relevant work under SRPT-n to the relevant work in a single-server system with server speed n under SRPT-1, using a known result (Lemma 2 in [17]). Third, we invoke Lemma C.1 to conclude.

We first define the following notation for work.

Definition A.3 (Relevant work under SRPT-n). Consider a tagged job with size x in an n-server system under SRPT-n. Define the relevant work of the tagged job, $W_x^{SRPT\text{-}n}$, to be the total work comprising jobs with remaining size smaller than x.

Definition A.4 ($W_{A_x}(t)$). Define $W_{A_x}(t)$ to be the total amount of work comprising of jobs arriving after time 0 and before time t with size smaller than x.

We can now bound the response time of a job by the above two terms of work.

Lemma A.5. *Without loss of generality, assume the tagged job arrives at time 0. For the tagged job with size x, we have that*

$$\mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] \leq \mathbb{P}\left[x + \frac{1}{n}W_x^{SRPT\text{-}n} + \frac{1}{n}W_{A_x}(t) > t\right]. \tag{32}$$

Proof. By PASTA (Poisson Arrivals See Time Averages), when the tagged job arrives, it sees $W_x^{SRPT\text{-}n}$ relevant work in the system. Under SRPT-n, the total work served before the tagged job is completed is upper bounded by: (1) the initial relevant work $W_x^{SRPT\text{-}n}$, (2) the x work of the tagged job itself, and (3) the total work from jobs with size smaller than x arriving before the tagged job is completed. The relevant work, together with newly arriving work comprising jobs with size smaller than x, must be completed with rate at least n, unless there are fewer than n jobs with remaining size smaller than the tagged job and the tagged job is served. The response time of the tagged job can be upper bounded by the above argument.

Specifically, define a potential function $G(s)$ at time s to be the sum of the following three terms: (1) remaining size of the tagged job, (2) $\frac{1}{n}$ times the remaining work from the initial relevant work $W_x^{SRPT\text{-}n}$, and (3) $\frac{1}{n}$ times the remaining work from the newly arrived work $W_{A_x}(s)$.

Hence, we have that $G(0) = x + \frac{1}{n}W_x^{SRPT\text{-}n}$. The dynamics of $G(s)$ are given by: At arrivals, $G(s)$ increases by $\frac{1}{n}$ times the size of the arriving job if the arriving job is smaller than x. Between arrivals, $G(s)$ decreases with rate at least 1 as long as the tagged job is in the system (if the tagged job is not served, then all n servers are serving jobs with size smaller than x, each reducing the potential function with rate $\frac{1}{n}$; otherwise if the tagged job is served, then the first term of the potential function already decreases with rate 1).

Mathematically, integrating the decrease and adding the increases at arrivals, we have that for any s before the tagged job completes,

$$G(s) \leq x + \frac{1}{n}W_x^{SRPT\text{-}n} + \frac{1}{n}W_{A_x}(s) - s. \tag{33}$$

Consider the event that the tagged job is not completed before time t (which means its response time is more than t). In this case, $G(t) > 0$ since the tagged job still has positive remaining size, and (33) holds for $s = t$. Hence, we have that

$$0 < G(t) \leq x + \frac{1}{n}W_x^{SRPT\text{-}n} + \frac{1}{n}W_{A_x}(t) - t. \tag{34}$$

Hence we have that

$$\mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] \leq \mathbb{P}\left[x + \frac{1}{n}W_x^{SRPT\text{-}n} + \frac{1}{n}W_{A_x}(t) > t\right].$$

□

Now we relate the relevant work under SRPT-n to the relevant work in a single-server system under SRPT-1.

Lemma A.6 (Lemma 2 in [17]). *Given an n-server system under SRPT-n with Poisson arrivals and total load $\rho < 1$. Consider a single-server system under SRPT-1 with the same arrival process but the server has speed n. Define W_x^{super} to be the relevant work that a job with size x sees in this single-server system. Then we have that*

$$W_x^{SRPT\text{-}n} \leq W_x^{super} + nx. \tag{35}$$

Note that although it is assumed that the job size distribution has finite variance in [17], the proof for this lemma does not require this assumption. Moreover, Lemma 2 in [17] is stated for n servers of speed $\frac{1}{n}$ each and a single server of speed 1; since it holds for any arrival sequence, rescaling time by a factor of n gives Lemma A.6.

For simplicity, now consider a single-server system with server speed 1 but all job sizes (and consequently the work) are scaled by $\frac{1}{n}$. Define the scaled notation $W_x^{SRPT-1} := \frac{1}{n} W_x^{super}$ and $W_{A_x}^{SRPT-1}(t) := \frac{1}{n} W_{A_x}(t)$. This allows us to use the known result for single-server systems under SRPT-1, see Lemma C.1.

We are now ready to prove Lemma A.2.

Proof of Lemma A.2. Let C be the constant from Lemma C.1 and consider a tagged job with size x. By Lemma A.5 with $t = Cx$,

$$\mathbb{P}\left[T_x^{SRPT\text{-}n} > Cx\right] \leq \mathbb{P}\left[x + \frac{1}{n} W_x^{SRPT\text{-}n} + \frac{1}{n} W_{A_x}(Cx) > Cx\right]. \tag{36}$$

By Lemma A.6, $W_x^{SRPT\text{-}n} \leq W_x^{super} + nx$. Substituting and using $W_x^{SRPT-1} = \frac{1}{n} W_x^{super}$ and $W_{A_x}^{SRPT-1}(Cx) = \frac{1}{n} W_{A_x}(Cx)$, we obtain

$$\begin{aligned}
&\mathbb{P}\left[x + \frac{1}{n} W_x^{SRPT\text{-}n} + \frac{1}{n} W_{A_x}(Cx) > Cx\right] \\
&\leq \mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(Cx) > Cx\right] \\
&= o(x^{-\beta}). && \text{Lemma C.1}
\end{aligned}$$

□

We can now prove Theorem A.1.

Proof of Theorem A.1. Pick $\beta > \alpha$ and let C be the constant from Lemma A.2. Then we have that

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{SRPT\text{-}n} > t\right]}{\mathbb{P}\left[S > t\right]} \\
&= \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \int_0^\infty \mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] \cdot f_S(x)dx \\
&= \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \left(\mathbb{P}\left[S > t\right] + \int_0^{t/C} \mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] \cdot f_S(x)dx + \int_{t/C}^{t} \mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] f_S(x)dx \right) \\
&\le 1 + \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \left(\int_0^{t/C} \mathbb{P}\left[T_{t/C}^{SRPT\text{-}n} > t\right] \cdot f_S(x)dx + \int_{t/C}^{t} 1 \cdot f_S(x)dx \right) \\
&\le 1 + \lim_{t\to\infty} \frac{\mathbb{P}\left[T_{t/C}^{SRPT\text{-}n} > t\right]}{\mathbb{P}\left[S > t\right]} \cdot \mathbb{P}\left[S \le t/C\right] + \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \cdot \mathbb{P}\left[S > t/C\right] \\
&\le 1 + \lim_{t\to\infty} \frac{\mathbb{P}\left[T_{t/C}^{SRPT\text{-}n} > t\right]}{\mathbb{P}\left[S > t/C\right]} \cdot \frac{\mathbb{P}\left[S > t/C\right]}{\mathbb{P}\left[S > t\right]} \cdot 1 + \lim_{t\to\infty} \frac{1}{\mathbb{P}\left[S > t\right]} \cdot \mathbb{P}\left[S > t/C\right] \\
&= 1 + 0 + C^\alpha.
\end{aligned}$$

Here the last equation uses Lemma A.2. □

B SRPT-n is not strongly tail optimal

In this section, we prove that SRPT-n is not strongly tail optimal for any $n \ge 2$ and any load $\rho < 1$ (Theorem B.1), complementing Theorem A.1. We first show the intuitions and then the formal proofs.

Intuitive arguments: Roughly speaking, Lower bound 1 and 2 in Section 3.2 say that no policy can complete a very large job faster than rate $\min\{1, n(1-\rho)\}$, and which of the two terms is smaller depends on the load. When $\rho < \frac{n-1}{n}$, we have $n(1-\rho) > 1$, and Lower bound 2 is tighter; when $\rho \ge \frac{n-1}{n}$, we have $n(1-\rho) \le 1$, and Lower bound 1 is tighter, which means, roughly, that the very large jobs need to run with rate $n(1-\rho)$ on average.

Under SRPT-n, a very large job has lower priority than all other jobs for almost all of its service, so it is served only when the other jobs leave at least one server idle, and then on a single server. Let N denote the steady-state number of jobs in an SRPT-n system, and define

$$q := \mathbb{P}\left[N < n\right],$$

the fraction of time during which at least one server is idle. The very large job is then completed at rate roughly q, which we will show is strictly smaller than $\min\{1, n(1-\rho)\}$ under both loads.

When $\rho < \frac{n-1}{n}$, the relevant comparison is $q < 1$: Completing the very large job at rate 1 would require serving it all the time, but under SRPT-n it waits whenever the other jobs occupy all n servers, which happens a positive fraction of the time. When $\rho \ge \frac{n-1}{n}$, the relevant comparison is $q < n(1-\rho)$: Here $n(1-\rho)$ is the average number of idle servers, whereas the very large job uses at most one idle server at a time. Whenever two or more servers are idle at the same time, the capacity of all but one of them is lost, and this lost capacity is the reason why SRPT-n is not strongly tail optimal.

Formal arguments:

Theorem B.1. *Suppose $n \geq 2$ and $\rho < 1$. Then*

$$\liminf_{t\to\infty} \frac{\mathbb{P}\left[T^{SRPT\text{-}n} > t\right]}{\mathbb{P}\left[S > \min\{1, n(1-\rho)\}\, t\right]} \geq \left(\frac{\min\{1, n(1-\rho)\}}{q}\right)^{\alpha} > 1. \tag{37}$$

Consequently, SRPT-n is not strongly tail optimal.

The proof has three steps. First, Lemma B.2 shows that $q < \min\{1, n(1-\rho)\}$, which gives the strict inequality in (37). Second, Lemma B.3 shows that a very large job is indeed completed at rate at most roughly q, which gives the first inequality in (37). Third, to conclude that SRPT-n is not strongly tail optimal, we compare it with Thresh-SPLIT(d), whose tail gets arbitrarily close to $\mathbb{P}\left[S > \min\{1, n(1-\rho)\}t\right]$ by Theorem 5.4.

Lemma B.2. *Suppose $n \geq 2$ and $\rho < 1$. Then $0 < q < \min\{1, n(1-\rho)\}$.*

Proof. Under SRPT-n, exactly $\min\{N, n\}$ servers are busy, and in steady state the rate at which work is completed equals the rate $\lambda \mathbf{E}[S] = n\rho$ at which work arrives. Hence $\mathbf{E}[\min\{N, n\}] = n\rho$, and the average number of idle servers is

$$n(1-\rho) = \mathbf{E}\left[(n-N)^+\right] = \mathbb{P}[N < n] + \mathbf{E}\left[(n-N-1)^+\right] = q + \mathbf{E}\left[(n-N-1)^+\right],$$

where $z^+ := \max\{z, 0\}$, and the second equality holds because $(n-N)^+ = \mathbf{1}\{N < n\} + (n-N-1)^+$. A stable M/G/$n$ system is empty with positive probability, so $\mathbf{E}\left[(n-N-1)^+\right] \geq (n-1)\mathbb{P}[N=0] > 0$, where we use $n \geq 2$. Hence $q < n(1-\rho)$, and $q \geq \mathbb{P}[N=0] > 0$. Finally, $\mathbb{P}[N \geq n] > 0$, since with positive probability n jobs arrive to an empty system within ϵ time units and each of them has size larger than ϵ, for any $\epsilon > 0$ with $\mathbb{P}[S > \epsilon] > 0$. Hence $q < 1$. □

Next, we show that a very large job is roughly completed at rate at most q. The difficulty is that at some point, the very large job will get high priority: Once its remaining size is small, it overtakes other jobs. The proof of the next lemma therefore only tracks the service needed to reduce its remaining size to a small fraction δ of its size.

Lemma B.3. *Suppose $n \geq 2$ and $\rho < 1$. Then for any constant $c > q$,*

$$\lim_{x\to\infty} \mathbb{P}\left[T_x^{SRPT\text{-}n} > \frac{x}{c}\right] = 1.$$

Proof. Consider a tagged job of size x that arrives at time 0, and consider an auxiliary SRPT-n system with the same jobs present at time 0^- and the same arrivals after time 0, but without the tagged job. By PASTA, as in the proof of Lemma 4.7, the auxiliary system is in steady state at time 0, and its state does not depend on x. Let $N(s)$ be the number of jobs in the auxiliary system at time s, and let

$$A(u) := \int_0^u \mathbf{1}\{N(s) < n\}\, ds$$

be the total time during $[0, u]$ at which the auxiliary system has at least one idle server. By the ergodic theorem, $\frac{A(u)}{u} \to q$ almost surely as $u \to \infty$.

Fix $c > q$ and choose $\delta \in (0, 1)$ such that $\frac{q}{c} < 1 - \delta$. Consider the following three events:

(1) Every job present at time 0^- has remaining size at most δx.
(2) Every job arriving during $(0, \frac{x}{c}]$ has size at most δx.
(3) $A(\frac{x}{c}) < (1-\delta)x$.

We first show that if all three events occur, then $T_x^{SRPT\text{-}n} > \frac{x}{c}$. By events 1 and 2, every job other than the tagged job has remaining size at most δx during $[0, \frac{x}{c}]$. Hence, as long as the remaining size of the tagged job is larger than δx, every other job has higher priority than the tagged job. Since SRPT-n serves the n jobs with the smallest remaining sizes, the tagged job then only occupies

a server that is idle in the auxiliary system: The other jobs receive exactly the same service as in the auxiliary system, and the tagged job is served, on a single server, exactly when $N(s) < n$. To reduce its remaining size from x to δx, the tagged job needs $(1-\delta)x$ units of service, but by event 3, it receives less than $(1-\delta)x$ units of service by time $\frac{x}{c}$. Hence, at time $\frac{x}{c}$, its remaining size is still larger than δx, and in particular it has not completed.

It remains to show that each of the three events occurs with probability tending to 1 as $x \to \infty$. First, the largest remaining size among the jobs present at time 0^- is finite almost surely, and its distribution does not depend on x, so event 1 occurs with probability tending to 1. Second, jobs with size larger than δx arrive according to a Poisson process with rate $\lambda \mathbb{P}[S > \delta x]$, so

$$\mathbb{P}[\text{event 2}] = \exp\left(-\lambda \frac{x}{c} \mathbb{P}[S > \delta x]\right) \to 1,$$

where we use $\delta x \, \mathbb{P}[S > \delta x] \le \mathbf{E}[S \mathbf{1}\{S > \delta x\}] \to 0$, since $\mathbf{E}[S] < \infty$.

Third, almost surely,

$$\frac{A(x/c)}{x} = \frac{1}{c} \cdot \frac{A(x/c)}{x/c} \to \frac{q}{c} < 1 - \delta.$$

So event 3 occurs with probability tending to 1. By a union bound over the complements of the three events, all three events occur with probability tending to 1, which proves the lemma. □

We are now ready to prove Theorem B.1.

Proof of Theorem B.1. Fix a constant $c > q$. For a tagged job of size $x > ct$, we have $\frac{x}{c} > t$, so $\mathbb{P}\left[T_x^{SRPT\text{-}n} > t\right] \ge \mathbb{P}\left[T_x^{SRPT\text{-}n} > \frac{x}{c}\right]$. Hence

$$\mathbb{P}\left[T^{SRPT\text{-}n} > t\right] \ge \mathbb{P}\left[T^{SRPT\text{-}n} > t,\ S > ct\right] \ge \left(\inf_{x > ct} \mathbb{P}\left[T_x^{SRPT\text{-}n} > \frac{x}{c}\right]\right) \mathbb{P}[S > ct],$$

where the infimum tends to 1 as $t \to \infty$ by Lemma B.3. Dividing by $\mathbb{P}[S > \min\{1, n(1-\rho)\}t]$ and using (1), we have that

$$\liminf_{t\to\infty} \frac{\mathbb{P}\left[T^{SRPT\text{-}n} > t\right]}{\mathbb{P}[S > \min\{1, n(1-\rho)\}t]} \ge \lim_{t\to\infty} \frac{\mathbb{P}[S > ct]}{\mathbb{P}[S > \min\{1, n(1-\rho)\}t]} = \left(\frac{\min\{1, n(1-\rho)\}}{c}\right)^{\alpha}.$$

Letting $c \downarrow q$ gives the first inequality in (37), and Lemma B.2 gives the second.

It remains to exhibit a policy whose tail is strictly lighter than that of SRPT-n. By the proof of Theorem 5.4, as the threshold d increases to ∞ (if $\rho \le \frac{n-1}{n}$) or to d^* (if $\rho > \frac{n-1}{n}$), the small-job system of Thresh-SPLIT(d) remains stable and $1 - r_{>d}$ increases to $\min\{1, n(1-\rho)\}$. Since $\min\{1, n(1-\rho)\} > q$, we can choose such a threshold d with $1 - r_{>d} > q$. By Theorem 5.3, (37), and (1),

$$\liminf_{t\to\infty} \frac{\mathbb{P}\left[T^{SRPT\text{-}n} > t\right]}{\mathbb{P}\left[T^{\text{Thresh-SPLIT}(d)} > t\right]} = \liminf_{t\to\infty} \frac{\mathbb{P}\left[T^{SRPT\text{-}n} > t\right]}{\mathbb{P}[S > (1 - r_{>d})t]} \ge \left(\frac{1 - r_{>d}}{q}\right)^{\alpha} > 1.$$

Hence, the supremum in the definition of tail optimality (Section 1.1) is strictly larger than 1 for SRPT-n, so SRPT-n is not strongly tail optimal. □

C Results for single-server systems under SRPT-1

The goal of this appendix is to prove the following lemma for single-server systems under SRPT-1 policy:

Lemma C.1. *For a single-server system under SRPT-1 policy with load $\rho < 1$, we have that for any $\beta > 0$, there exists a constant $C > 0$ such that*

$$\mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(Cx) > Cx\right] = o(x^{-\beta}). \tag{38}$$

The proof follows a conventional method for single-server systems under SRPT-1 (e.g., see the proof of Lemma 18 in [27]). It uses the following lemma, a standard result for random walks (see, e.g., Lemma 16 in [27], which is due to [29]).

Lemma C.2 (Lemma 16 in [27]). *Let X_i be a sequence of i.i.d. random variables with $\mathbf{E}[X_i] < 0$ and $\mathbf{E}\left[(X_i^+)^p\right] < \infty$ for some $p > 1$. Let $\mathcal{S}_n(x) := \sum_{i=1}^n (\min\{X_i, x\})$ for some $x > 0$. Define $M(x) := \sup_n \mathcal{S}_n(x)$. Then, for any $0 < \beta < \infty$, there exist constants $k > 0$ such that*

$$\mathbb{P}[M(x) > kx] = o(x^{-\beta}).$$

Proof of Lemma C.1. Define S_i to be the size of the ith arrival and A_i to be the ith interarrival time. Now define an auxiliary sample path with the same interarrival times ($A_i' := A_i$) but $S_i' := \min(S_i, x)$. Define $W_A'(t)$ to be the total work from jobs in the auxiliary sample path which arrive after time 0 and before time t. Then we have that $W_A'(t) \geq W_{A_x}^{SRPT-1}(t)$. Define ρ' to be the load of the auxiliary sample path, i.e., $\rho' := \frac{\mathbf{E}[S_i']}{\mathbf{E}[A_i']} < 1$.

Define W_x' to be the steady-state work in the auxiliary sample path. Now we prove that $W_x' + x \geq W_x^{SRPT-1}$: In the original sample path, denote the consecutive period of time where $W_x^{SRPT-1} > 0$ by "x busy period". Then, during each x busy period, W_x^{SRPT-1} only increases when a job with size smaller than x arrives, while W_x' increases when every job arrives (if the job size is larger than x, W_x' increases by x). Hence, we have that the difference $W_x' - W_x^{SRPT-1}$ never decreases during an x busy period. Moreover, at the start of each x busy period, $W_x^{SRPT-1} \leq x$. This gives the proof of the inequality $W_x' + x \geq W_x^{SRPT-1}$.

Define the auxiliary quantity

$$U_x^c := \sup_{t>0}[W_A'(t) - ct]. \tag{39}$$

By the Loynes construction [24], U_x^1 is the steady-state relevant work in the auxiliary sample path, i.e., $U_x^1 =_{st} W_x'$. Hence we have that $U_x^1 + x \geq_{st} W_x^{SRPT-1}$.

Pick $\delta < 1 - \rho'$ and let $C \geq \frac{12}{\delta}$. Then we have that

$$\begin{aligned}
&\mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(Cx) > Cx\right] \\
&\leq \mathbb{P}\left[3x + U_x^1 + W_A'(Cx) > Cx\right] \\
&= \mathbb{P}\left[U_x^1 + W_A'(Cx) - (1-\delta)Cx + 3x > \delta Cx\right] \\
&\leq \mathbb{P}\left[U_x^1 > \frac{\delta Cx}{2}\right] + \mathbb{P}\left[W_A'(Cx) - (1-\delta)Cx + 3x > \frac{\delta Cx}{2}\right].
\end{aligned}$$

For the first term, note that $U_x^1 \leq U_x^{1-\delta}$, hence

$$\mathbb{P}\left[U_x^1 > \frac{\delta Cx}{2}\right] \leq \mathbb{P}\left[U_x^{1-\delta} > \frac{\delta Cx}{2}\right].$$

For the second term, note that $W_A'(Cx) - (1-\delta)Cx \leq \sup_{t>0}[W_A'(t) - (1-\delta)t] = U_x^{1-\delta}$, hence we have that

$$\mathbb{P}\left[W_A'(Cx) - (1-\delta)Cx + 3x > \frac{\delta Cx}{2}\right] \leq \mathbb{P}\left[U_x^{1-\delta} > \left(\frac{\delta C}{2} - 3\right)x\right] \leq \mathbb{P}\left[U_x^{1-\delta} > \frac{\delta Cx}{4}\right].$$

Hence,

$$\mathbb{P}\left[2x + W_x^{SRPT-1} + W_{A_x}^{SRPT-1}(Cx) > Cx\right] \leq \mathbb{P}\left[U_x^{1-\delta} > \frac{\delta Cx}{2}\right] + \mathbb{P}\left[U_x^{1-\delta} > \frac{\delta Cx}{4}\right] \leq 2\mathbb{P}\left[U_x^{1-\delta} > \frac{\delta Cx}{4}\right]. \tag{40}$$

It remains to show $\mathbb{P}\left[U_x^{1-\delta} > \frac{\delta C x}{4}\right] = o(x^{-\beta})$. Because the supremum in $U_x^{1-\delta}$ is attained at arrival instants, we can write $U_x^{1-\delta} = \sup_n \sum_{i=1}^n [S_i' - (1-\delta)A_i']$. Because $1-\delta > \rho'$, the step sizes $X_i := S_i' - (1-\delta)A_i'$ have negative mean. Moreover, each S_i' is bounded above by x, so $\min\{X_i, x\} = X_i$. Applying Lemma C.2 with $M(x) = U_x^{1-\delta}$, for any $\beta > 0$, there exists a constant $\bar{k} > 0$ such that

$$\mathbb{P}\left[U_x^{1-\delta} > \bar{k}x\right] = o(x^{-\beta}).$$

Setting $C = \max\{\frac{4\bar{k}}{\delta}, \frac{12}{\delta}\}$ completes the proof. □

D Proof of Theorem 6.2

Theorem 6.2. *Under policy TAGS-SPLIT(d) and Approximation 1, if* $r_{small} := \rho \cdot n - \rho_{large}^{TAGS}(d) < n-1$ *(i.e., the small-job system is stable), then*

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{TAGS\text{-}SPLIT(d)} > t\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t\right]} = 1. \tag{28}$$

Proof. **Small jobs.** For jobs with size $S \le d$, their response time under TAGS-SPLIT(d) is that of a FCFS system with $n-1$ servers and job sizes bounded by d. By Lemma 5.2, there exists a constant $\gamma > 0$ such that

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{TAGS-SPLIT}(d)} > t \mid S \le d\right]}{e^{-\gamma t}} \le 1. \tag{41}$$

Big jobs. For jobs with size $S > d$, the response time under TAGS-SPLIT(d) consists of three parts: (i) the waiting time in the FCFS queue, denoted by T_{wait}^{FCFS}, (ii) d units of service at the FCFS servers, and (iii) the time at the PS server, denoted by T_{PS}. Part (i) decays exponentially by Lemma 5.2, and part (ii) is a constant. Part (iii) is the response time of a single-server PS system with load $\rho_{large}^{TAGS}(d)$ and job size distribution $(S-d \mid S > d)$. By Approximation 1 and [19], the PS tail satisfies

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T_{PS} > t\right]}{\mathbb{P}\left[S-d > (1-\rho_{large}^{TAGS}(d))t \mid S > d\right]} = 1. \tag{42}$$

Hence we have that the tail of the response time of a big job under TAGS-SPLIT(d) satisfies

$$\begin{aligned}
\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{TAGS-SPLIT}(d)} > t \mid S > d\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t \mid S > d\right]} &= \lim_{t\to\infty} \frac{\mathbb{P}\left[T_{wait}^{FCFS} + d + T_{PS} > t\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t \mid S > d\right]} \\
&= \lim_{t\to\infty} \frac{\mathbb{P}\left[T_{PS} > t\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t \mid S > d\right]} \\
&= \lim_{t\to\infty} \frac{\mathbb{P}\left[T_{PS} > t\right]}{\mathbb{P}\left[S-d > (1-\rho_{large}^{TAGS}(d))t \mid S > d\right]} \\
&= 1,
\end{aligned} \tag{43}$$

where the second equality uses that T_{wait}^{FCFS} has exponentially decaying tail (part (i)) and d is a constant (part (ii)), both negligible compared to the regularly varying T_{PS};[2] the third uses that

[2]This argument can be made rigorous trivially by splitting t into ϵt and $(1-\epsilon)t$, and then using a union bound to bound both terms.

$\mathbb{P}[S-d>x \mid S>d] \sim \mathbb{P}[S>x \mid S>d]$ as $x \to \infty$ for regularly varying S; and the last equality uses (42).

Combining. By the law of total probability,

$$\lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{TAGS-SPLIT}(d)} > t\right]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t\right]}$$
$$= \lim_{t\to\infty} \frac{\mathbb{P}\left[T^{\text{TAGS-SPLIT}(d)} > t \mid S \le d\right]\mathbb{P}[S\le d] + \mathbb{P}\left[T^{\text{TAGS-SPLIT}(d)} > t \mid S > d\right]\mathbb{P}[S>d]}{\mathbb{P}\left[S > (1-\rho_{large}^{TAGS}(d))t\right]}.$$

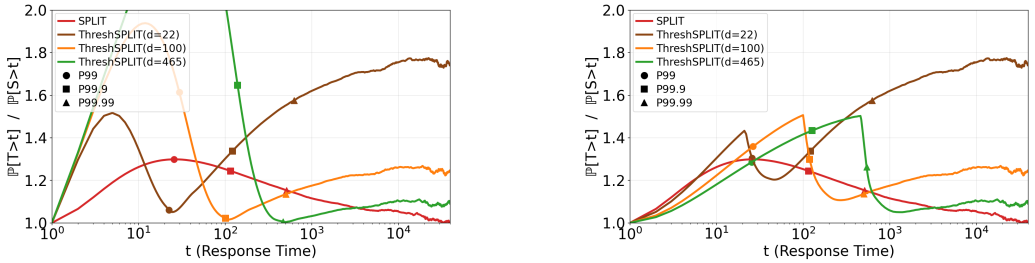
The first term vanishes because the exponential decay in (41) is faster than any heavy-tailed distribution. The second term converges to 1 by (43). □

E Additional simulation results

E.1 Thresh-SPLIT(d) simulation under different scheduling policies for the small jobs

As discussed in Sections 5 and 8.1, the choice of scheduling policy on the $n-1$ small-job servers in Thresh-SPLIT(d) is not essential: because small jobs' sizes are bounded by d, any reasonable policy conceivably yields an exponentially decaying response time tail for small jobs (Lemma 5.2), so the asymptotic tail of Thresh-SPLIT(d) matches Theorem 5.4 under either FCFS or SRPT-$(n-1)$. In Section 7, the low-load figures (Section 7.1) use FCFS on the small-job servers, while the high-load figures (Section 7.2) use SRPT-$(n-1)$; the reason is that FCFS produces a visibly larger bump in the pre-asymptotic regime at high load. In this appendix, we show the two variants side by side on the three simulation settings of Section 7 to validate both points empirically.

Across all three settings, the asymptotic behavior of the two variants coincides, matching our theoretical prediction. In the pre-asymptotic regime, the two choices give similar curves at low load (Figure 10), while FCFS exhibits a visibly larger bump at high load (Figures 11 and 12).



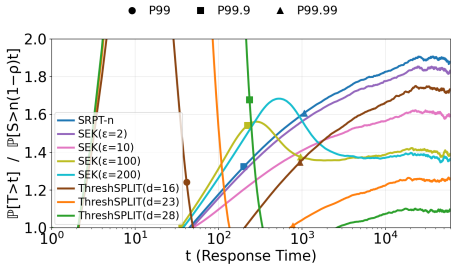
(a) FCFS for small jobs

(b) SRPT-$(n-1)$ for small jobs

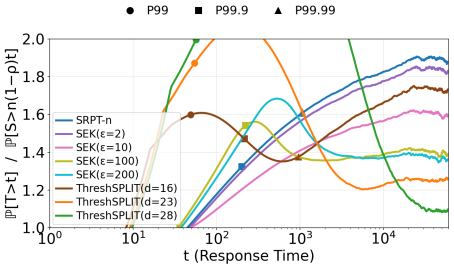
Fig. 10. Thresh-SPLIT(d), low-load setting ($n=3$, $\rho=0.5$, $\alpha=1.5$).

E.2 Simulation figures with SEK policies for different parameters

In Section 7, the figures in the main text show only the SEK-ϵ curve with the largest ϵ in our sweep, since this gives the best asymptotic tail among the SEK family. For completeness, this appendix replicates Figures 6a, 8a, 8b, 7a, and 7b with the full set of SEK-ϵ curves for several values of ϵ. Figure 13 shows the low-load setting, Figure 14 the high-load settings, and Figure 15 the varying

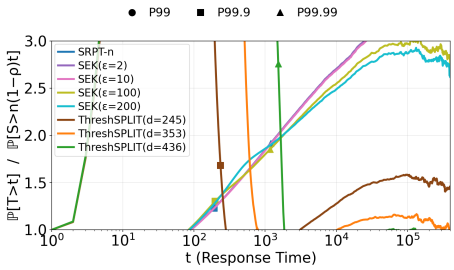


(a) FCFS for small jobs

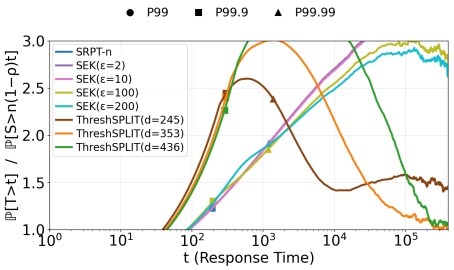


(b) SRPT-$(n-1)$ for small jobs

Fig. 11. Thresh-SPLIT(d), high-load setting ($n = 3$, $\rho = 0.8$, $\alpha = 1.5$).



(a) FCFS for small jobs



(b) SRPT-$(n-1)$ for small jobs

Fig. 12. Thresh-SPLIT(d), higher-load setting ($n = 10$, $\rho = 0.94$, $\alpha = 1.5$).

tail index α. Across all settings, we observe that SEK-ϵ improves on SRPT-n as ϵ grows, but the asymptotic improvement saturates.

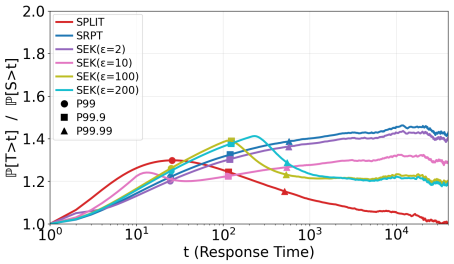
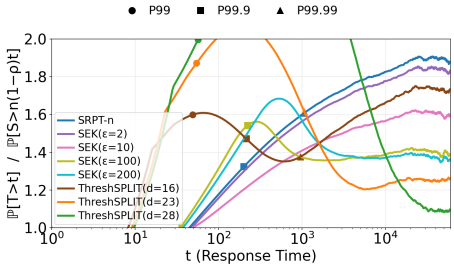
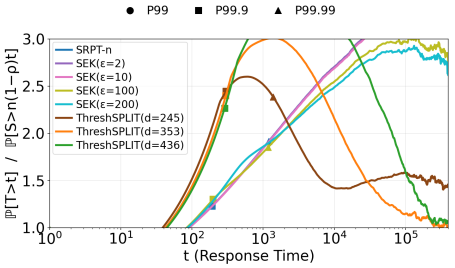


Fig. 13. Low-load setting ($n = 3$, $\rho = 0.5$, $\alpha = 1.5$); cf. Figure 6a.

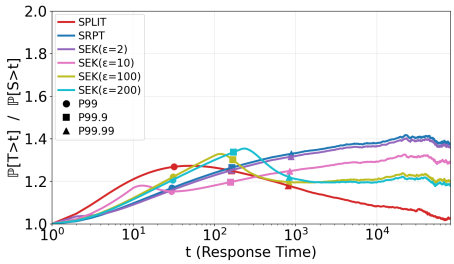


(a) $n = 3$, $\rho = 0.8$; cf. Figure 8a.

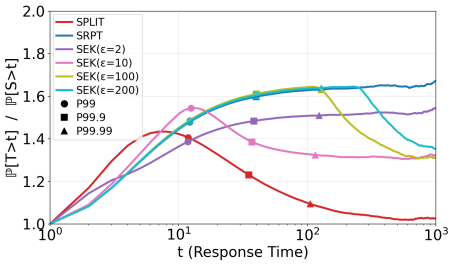


(b) $n = 10$, $\rho = 0.94$; cf. Figure 8b.

Fig. 14. High-load settings, Pareto job sizes with $\alpha = 1.5$.



(a) $\alpha = 1.4$; cf. Figure 7a.



(b) $\alpha = 2.0$; cf. Figure 7b.

Fig. 15. Varying Pareto tail index, $n = 3$ servers, $\rho = 0.5$.