Reducing Heavy-Traffic Response Time with Asymmetric Dispatching

Runhan Xie
University of California, Berkeley
runhan_xie@berkeley.edu

Ziv Scully
Cornell University
zivscully@cornell.edu

1. INTRODUCTION

Reducing mean response time has always been a desirable goal in queueing systems. If job sizes (a.k.a. service times) are known to the scheduler, the policy that minimizes mean response time of a single-server queue is SRPT (Shortest Remaining Processing Time). This is true even for queues that are part of a larger system, such as immediate-dispatch systems where jobs are sent to one of multiple single-server queues upon arrival.

Despite the optimality of SRPT, in practice, FCFS (First-Come, First-Served) is very common, even when job sizes are known or can be estimated well. Concerns about SRPT include added implementation complexity, potential overheads due to preempting jobs, and possible unfairness towards large jobs. In light of this, we ask:

> With known job sizes, what routing policy minimizes mean response time in immediate-dispatch systems with FCFS queues? Can we match the performance of SRPT queues?

Hyytiä et al. [8] identify this as an open problem, even in very simple settings like the following:

- We dispatch to just two identical single-server queues, both using FCFS.
- We have just two possible job sizes: short jobs of size a, and long jobs of size b.

We take a first attempt towards solving this problem by considering it with M/G arrivals in the *heavy-traffic regime*, i.e. as the system load approaches capacity (see Section 2).

1.1 Baselines: LWL and SITA

A number of dispatching policies have been studied with FCFS servers. Many of these are designed for limited information settings, such as unknown job sizes or incomplete information about each servers' state. But with known job sizes, it is reasonable to assume we know the amount of work at each server at every moment in time. The two dispatching policies from the literature that make the most sense in our setting are therefore *LWL* (Least-Work-Left) and *SITA* (Size-Interval-Task-Assignment) [4].

- LWL dispatches each job to the server with least work present when the job arrives.
- SITA splits job sizes into intervals, then dispatches jobs in each size interval to a different server.

It is known that LWL and SITA can each sometimes outperform the other, including in heavy traffic [5]. Moreover, neither matches the performance achievable using SRPT (see Fig. 2), suggesting there may be room to improve.

Other more complicated dispatching heuristics have been proposed in recent work [6, 7]. They show promise in simulation, but analyzing them theoretically, even in heavy traffic, remains open. We thus focus on LWL and SITA as baselines.

1.2 Our Contribution

Our main contribution is a new dispatching policy called CRAB (**C**areful-**R**outing-to-**A**chieve-**B**ound). CRAB outperforms both SITA and LWL in heavy traffic. In the case where the load of the short jobs is less than half of the total service capacity, it even matches resource-pooled SRPT, which means it is heavy-traffic optimal (see Fig. 2).

In the rest of this abstract, we:

- Define our model and the CRAB policy (Section 2).
- Compare the performance of CRAB to LWL, SITA, and resource-pooled SRPT (Section 3).
- Outline how we analyze CRAB's performance, including the obstacles we encounter and the ideas we use to overcome them (Section 4).

2. SYSTEM MODEL

We consider a queueing system with two identical FCFS servers, each having processing rate $1/2$ (i.e. a job of size s has processing time $2s$) and keeping its own queue. We have M/G arrivals: jobs arrive as a Poisson process with rate λ, and there is a two-point job size distribution

$$S = \begin{cases} a & \text{w.p. } q_a \\ b & \text{w.p. } q_b, \end{cases}$$

where $a < b$. We dispatch jobs immediately upon arrival. Let

$$\rho_a = \lambda a q_a \quad \text{and} \quad \rho_b = \lambda b q_b$$

be the loads due short and long jobs, respectively, and let $\rho = \rho_a + \rho_b$ be the total load. We focus on the heavy-traffic regime $\lambda \to 1/\mathbb{E}[S]$, which we denote by $\rho \uparrow 1$ for short. We further assume that $\rho_a \neq \rho_b$ for technical reasons, but we believe this can be relaxed.

We represent the system's state as a vector (W_l, W_s), where each element is the amount of work at one of the queues. The subscripts stand for "long" and "short", respectively, as we discuss when defining CRAB below. Abusing notation slightly, we write (W_l, W_s) to refer to both a generic system state and the stationary distribution of the system state.

The *CRAB* dispatching policy works by designating one server to maintain less average work in steady state. We will henceforth refer to this server as the *short server* (work W_s) and the other one as the *long server* (work W_l). CRAB has a

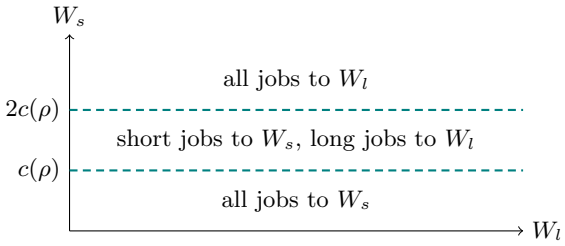

Figure 1: Illustration of the CRAB policy.

single threshold parameter, denoted $c(\rho)$. Roughly speaking, CRAB tries to keep W_s between $c(\rho)$ and $2c(\rho)$. It does so by as follows, as illustrated in Fig. 1.

- If $W_s \leqslant c(\rho)$, all jobs are dispatched to the short server.
- If $c < W_s \leqslant 2c(\rho)$, only short jobs are dispatched to the short server queue. All long jobs are dispatched to the long server queue.
- If $W_s > 2c(\rho)$, all jobs are dispatched to the long server queue until W_s drops below $2c(\rho)$.

3. CRAB VS. LWL, SITA, AND SRPT

In this section, we compare the heavy-traffic performance of CRAB to that of several other policies. Throughout this section, we write $\mathbb{E}[T_\pi]$ for the *mean response time*, namely the average amount of time between a job's arrival and departure, under policy π.

We begin by discussing LWL. It is known that for our job size distribution S the mean response time of LWL, denoted $\mathbb{E}[T_{\text{LWL}}]$ (and more generally $\mathbb{E}[T_\pi]$ for policy π) scales as

$$\mathbb{E}[T_{\text{LWL}}] \sim \frac{\frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]}}{1-\rho}, \quad \text{that is,} \quad \lim_{\rho\uparrow 1}(1-\rho)\mathbb{E}[T_{\text{LWL}}] = \frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]}.$$

We will use LWL as a baseline against which we compare other policies. As such, we define the *heavy-traffic constant* of policy π to be

$$K_\pi = \lim_{\rho\uparrow 1}\frac{\mathbb{E}[T_\pi]}{\mathbb{E}[T_{\text{LWL}}]}, \quad \text{so} \quad \mathbb{E}[T_\pi] \sim K_\pi \mathbb{E}[T_{\text{LWL}}].$$

Our main result characterizes CRAB's heavy-traffic constant.

THEOREM 3.1. *In the $\rho\uparrow 1$ limit, the CRAB policy with threshold $c(\rho)$ such that $\omega\big(\log\frac{1}{1-\rho}\big) < c(\rho) < o\big(\frac{1}{1-\rho}\big)$ has mean response time $\mathbb{E}[T_{\text{CRAB}}] \sim K_{\text{CRAB}}\mathbb{E}[T_{\text{LWL}}]$, where*

$$K_{\text{CRAB}} = \max\left\{\frac{\mathbb{E}[S]}{b}, 2-\frac{\mathbb{E}[S]}{a}\right\}.$$

We compare the heavy-traffic constant of CRAB to that of several other policies in Fig. 2. These include LWL, SITA,[1] and *resource-pooled* SRPT. This last policy is SRPT in a single $M/G/1$ with processing rate 1 (as opposed to our two rate-1/2 servers), without any dispatching.[2] One can straightforwardly show that with this resource pooling, $\mathbb{E}[T_{\text{SRPT}}]$ is a lower bound on the mean response time of any dispatching policy in our FCFS setting. Remarkably, CRAB matches SRPT's heavy-traffic performance when $\rho_a < \rho_b$.

[1]SITA is traditionally defined using a strict size threshold [4]. However, because we have only two job sizes, a strict threshold can result in one server becoming overloaded. As such, when $\rho_a > \rho_b$, we route a fraction of short jobs to the large jobs' server, and vice versa when $\rho_a < \rho_b$. We choose the exact fractions to minimize K_{SITA}, which turns out to very slightly bias the load towards the short jobs' server.

[2]In our setting, one can match the heavy-traffic performance of resource-pooled SRPT by dispatching to two SRPT servers with processing speed 1/2 [1].

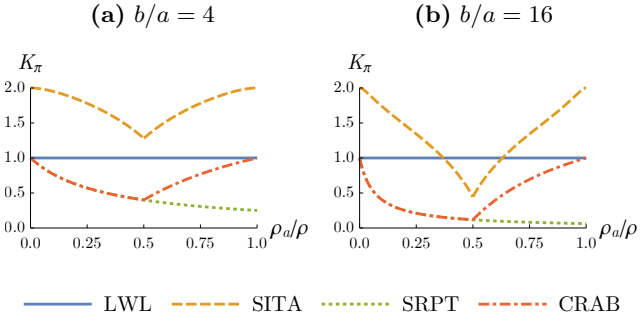

Figure 2: Comparing the heavy-traffic performance of CRAB against that of LWL, SITA, and (resource-pooled) SRPT. For each policy π, we show K_π, the number such that $\mathbb{E}[T_\pi] \sim K_\pi\mathbb{E}[T_{\text{LWL}}]$ as $\rho\uparrow 1$, as a function of ρ_a/ρ, the fraction of load consisting of small jobs, for two values of a/b, the ratio of large jobs' size to small jobs' size. CRAB always outperforms LWL and SITA, matching the lower bound of SRPT when $\rho_a/\rho < 1/2$.

3.1 Why CRAB Outperforms LWL and SITA

As has been previously observed by Harchol-Balter et al. [5], neither LWL nor SITA consistently outperforms the other. This is because the policies have different pros and cons.

- LWL actively regulates the system state, ensuring both servers have roughly equal amounts of work. However, that means both servers have a large amount of work, so all jobs have large response time.
- SITA dispatches asymmetrically, sending short jobs to a server which has less work on average. However, it does not actively regulate the system state, so short jobs can still experience large response times.

CRAB improves upon LWL and SITA by combining their pros. Like LWL, CRAB actively regulates the system state. But like SITA, it keeps one of the servers shorter than the other, allowing it to give short jobs a "fast path" with much lower response time. As we outline in Section 4, jobs sent to the short server have response time $O(c(\rho))$, which is negligible in heavy traffic as long as $c(\rho) < o\big(\frac{1}{1-\rho}\big)$.

We conclude this discussion by comparing CRAB to SRPT. In our system, $K_{\text{SRPT}} = \frac{\mathbb{E}[S]}{b}$. This matches K_{CRAB} when $\rho_a < \rho_b$, in which case CRAB is heavy-traffic optimal. The question remains whether CRAB is also heavy-traffic optimal when $\rho_a > \rho_b$. It seems difficult to match SRPT in this case, because we cannot send all the short jobs to one server while remaining stable in heavy traffic. But in heavy traffic, CRAB sends the minimum possible fraction of short jobs to the long server, even when $\rho_a > \rho_b$. We therefore conjecture CRAB is heavy-traffic optimal when $\rho_a > \rho_b$.

4. OBSTACLES TO FORMAL ANALYSIS

4.1 Stability

It is not immediately clear whether CRAB stabilizes the system, i.e. whether a stationary distribution exists. There is good reason to worry: if the threshold $c(\rho)$ is too small, e.g. if $c(\rho) = 0$, then the system can be unstable even if $\rho < 1$.

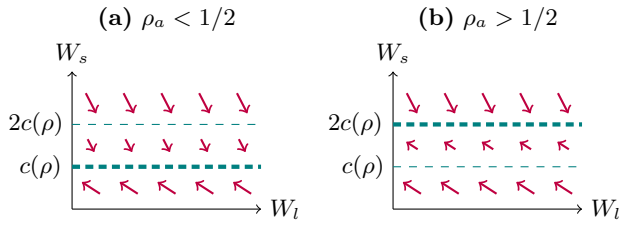

Figure 3: Average drift of the system state under CRAB in the $\rho_a < 1/2$ and $\rho_a > 1/2$ regimes. State space collapse occurs around the thick dashed lines.

How do we find a value of $c(\rho)$ that ensures stability? Usual approaches to proving stability, such as fluid limits and Foster-Lyapunov conditions, exploit the drift structure of the system. But as shown in Fig. 3, CRAB has complicated drift properties.

- There are discontinuities at $W_s = c(\rho)$ and $W_s = 2c(\rho)$, complicating the fluid limit approach.
- The long server is overloaded in some regions and underloaded in others, making it hard to find a Lyapunov function with negative drift outside a compact region.

To solve this issue, we used a hybrid approach. The key observation in our approach is that W_s is not impacted by W_l. We thus start by showing stability of W_s alone, for which standard Lyapunov functions suffice. With W_s in stationarity, the arrivals to the long server become a stationary process (albeit without i.i.d. interarrival times and job sizes). Loynes's construction for $G/G/1$ queues [9] therefore implies stability of W_l, provided the short server does not let through too much load.[3] Showing this is part of the next challenge.

4.2 State-Space Collapse of the Short Server

As shown in Fig. 3, W_s drifts towards the threshold $c(\rho)$ if $\rho_a < 1/2$ and towards the threshold $2c(\rho)$ if $\rho_a > 1/2$. As a result, we expect W_s to concentrate around the corresponding threshold with high probability. Specifically, we should expect *state-space collapse*, which is when the probability of W_s being further than distance d from the threshold decays exponentially in d. State-space collapse is important because it implies that the short server is rarely idle, which is important for both proving stability (Section 4.1) and bounding response time (Section 4.3).

The main theoretical tool used to prove state-space collapse is a generic result of Eryilmaz and Srikant [2, Lemma 1]. Roughly speaking, it says that state space collapse to a region occurs if the system drifts towards that region on average and has well controlled jumps. The result has been used for numerous heavy-traffic analyses. However, both the result and its uses are all for discrete-time systems, whereas we study a continuous-time system. While we could in principle consider an embedded discrete-time process, this quickly becomes cumbersome, particularly when verifying that jumps are well controlled.

Instead, we establish state-space collapse by proving a *new generic continuous-time state-space collapse result*. It works similarly to the result of Eryilmaz and Srikant [2], requiring average drift towards a region and well controlled jumps. We prove the result by applying Miyazawa's rate conservation law from Palm calculus [10].

[3] An alternative idea is to show that W_l satisfies a Foster-Lyapunov condition "on average" when W_s is in stationarity. Foss et al. [3] show how to do this, but in discrete time, so their results do not immediately apply to our setting.

4.3 Response Time Analysis

Under FCFS, a job's response time is the amount of work at the queue to which it is dispatched, so analyzing response time amounts to analyzing the joint distribution of (W_l, W_s).

State-space collapse implies that $W_s = \Theta(c(\rho)) < o\left(\frac{1}{1-\rho}\right)$ with high probability. In particular, the short server is rarely idle. This means the total work is similar to that of a resource-pooled $M/G/1$ queue, so $\mathbb{E}[W_l + W_s] = \Theta\left(\frac{1}{1-\rho}\right)$. This means W_l dominates W_s in heavy traffic, so only jobs dispatched to the long server contribute to the heavy-traffic constant K_{CRAB} (Section 3).

By the above discussion, in heavy traffic, a job's response time essentially boils down to two factors:

- the work at the long server W_l when it arrives, and
- whether it gets dispatched to the long server.

The main obstacle here is that these two factors are not independent! Specifically, both are dependent on W_s. We resolve this by showing that they are *approximately* independent. Formally, we show that if $\rho_a < 1/2$,

$$\left|\mathbb{E}[W_l \mid W_s < c(\rho)] - \mathbb{E}[W_l \mid W_s \geqslant c(\rho)]\right| \leqslant O(1), \quad (1)$$

and similarly for threshold $2c(\rho)$ if $\rho_a > 1/2$. This implies that all jobs dispatched to the long server see expected work approximately $\mathbb{E}[W_l]$.

Why should we expect (1) to hold? The key observation is that W_s cycles relatively rapidly between being above or below the threshold $c(\rho)$. Because the cycles are short, the change in W_l during a cycle is small, so the two conditional expectations in (1) cannot be too far apart. To formalize this intuition, we use the Palm inversion formula.

References

[1] D. G. Down and R. Wu. Multi-layered round robin routing for parallel servers. *Queueing Systems*, 53:177–188, 2006.

[2] A. Eryilmaz and R. Srikant. Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Systems*, 72:311–359, 2012.

[3] S. Foss, S. Shneer, and A. Tyurlikov. Stability of a markov-modulated markov chain, with application to a wireless network governed by two protocols. *Stochastic Systems*, 2(1): 208–231, 2013.

[4] M. Harchol-Balter, M. E. Crovella, and C. D. Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59 (2):204–228, 1999.

[5] M. Harchol-Balter, A. Scheller-Wolf, and A. R. Young. Surprising results on task assignment in server farms with high-variability workloads. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 287–298, 2009.

[6] E. Hyytiä and R. Righter. STAR and RATS: Multi-level dispatching policies. In *2020 32nd International Teletraffic Congress (ITC 32)*, pages 81–89. IEEE, 2020.

[7] E. Hyytiä and R. Righter. On sequential dispatching policies. In *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2022.

[8] E. Hyytiä, P. Jacko, and R. Righter. Routing with too much information? *Queueing Systems*, 100(3-4):441–443, 2022.

[9] R. M. Loynes. The stability of a queue with non-independent inter-arrival and service times. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 497–520. Cambridge University Press, 1962.

[10] M. Miyazawa. Rate conservation laws: a survey. *Queueing Systems*, 15:1–58, 1994.