A Little Redundancy Goes a Long Way: Convexity in Redundancy Systems

Kristen Gardner[a,*], Esa Hyytiä[b], Rhonda Righter[c]

[a] *Department of Computer Science, Amherst College, AC#2239, Amherst, MA 01002*
[b] *Department of Computer Science, University of Iceland, Dunhagi 5, 107 Reykjavk, Iceland*
[c] *Department of Industrial Engineering and Operations Research, UC Berkeley, 4141 Etcheverry Hall, Berkeley, CA 94720*

Abstract

Redundancy is an increasingly popular technique for reducing response times in computer systems, and there is a growing body of theoretical work seeking to analyze performance in systems with redundancy. The idea is to dispatch a job to multiple servers at the same time and wait for the first copy to complete service. Redundancy can help reduce response time because redundant jobs get to experience the shortest of multiple queueing times and potentially of multiple service times—but it can hurt jobs that are not redundant and must wait behind the redundant jobs' extra copies. Thus in designing redundancy systems it is critical to find ways to leverage the potential benefits without incurring the potential costs.

Scheduling represents one tool for maximizing the benefits of redundancy. In this paper we study three scheduling policies: First-Come First-Served (FCFS), Least Redundant First (LRF, under which less-redundant jobs have priority over more-redundant jobs), and Primaries First (PF, under which each job designates a "primary" copy, and all other copies have lowest priority). Our goal for each of these policies is to understand the marginal impact of redundancy: how much redundancy is needed to get the biggest benefit? We study this question analytically for LRF and FCFS, and via simulation for all three policies. One of our primary contributions is a surprisingly intricate proof that mean response time is convex as well as decreasing as the proportion of jobs that are redundant increases under LRF for exponential services. While response time under PF is also decreasing and appears to be convex as well, we find that, surprisingly, FCFS may be neither decreasing nor convex, depending on the parameter values. Thus, the scheduling policy is key in determining both whether redundancy helps and the marginal effects of adding more redundancy to the system.

Keywords: Redundancy, replication, scheduling, Least Redundant First, Primaries First

1. Introduction

A powerful tool for addressing the inherent variability and unreliability in cloud computing, mobile grids, volunteer desktop grids, and large-scale data access systems is *redundancy*, or the replication of jobs to multiple servers. The idea is to dispatch the job to several servers, where the job is considered complete as soon as *any* one of its copies completes; at this time all other copies are canceled immediately. This redundancy protocol is sometimes called "cancel-on-completion" in contrast to "cancel-on-start," where all copies are cancelled as soon as any copy starts service. Redundancy (also known as speculation, replication, or cloning) has led to significant improvements in response times and reliability in practice [7, 8, 15, 27]. However, it can be expensive to replicate data on multiple servers and to coordinate a large number of job replications, so it is generally not reasonable to copy all jobs on all servers. In this work, we study the marginal return of redundancy: can most of the benefit of redundancy be obtained with only a small amount of redundancy?

We model our system as a multi-server queueing model with general arrivals and exponential, server-dependent, service times. We assume some initial configuration of classes of jobs, where a class i is defined

*Corresponding author
Email addresses: kgardner@amherst.edu (Kristen Gardner), esa@hi.is (Esa Hyytiä), rrighter@berkeley.edu (Rhonda Righter)

by the set of servers S_i that can serve (copies of) jobs of that class. We assume the job classes follow a nested structure. That is, if two job classes i and j have at least one server in common ($S_i \cap S_j \neq \emptyset$), then one class is a subset of the other ($S_i \subset S_j$ or $S_j \subset S_i$). When a class-i job arrives to the system, a copy of the job is sent to all the servers in S_i; the first copy to complete completes the service of the job, and all other copies are removed without penalty.

This paper is a companion to our paper [16], in which we studied scheduling policies and fairness in nested systems with redundancy. We considered three scheduling policies: First Come First Served (FCFS), Least Redundant First (LRF), and Primaries First (PF). In LRF scheduling, jobs with smaller degrees of redundancy are given preemptive priority over more-redundant jobs at each server. We showed in [16] that LRF scheduling stochastically maximizes the departure process, and therefore minimizes overall mean response time, assuming Poisson arrivals and exponential service times. Unfortunately, redundancy is not always fair: under both LRF and FCFS—for which we derived exact, closed-form expressions for per-class and overall mean response time—mean response times for some classes can increase under FCFS and LRF scheduling relative to a system in which no jobs are redundant. We designed PF scheduling with fairness in mind; in particular, under PF no class of jobs is worse off when redundancy is introduced to the system. Under PF, each job designates one copy as its "primary," and all other copies (if any) are designated "secondaries." At each server, primaries are given full preemptive priority over secondaries; within the primaries (respectively, secondaries), jobs are scheduled in FCFS order. We showed in [16] that, under PF, the response time for each class is stochastically improved when some jobs shift from being non-redundant to redundant.

In this paper we consider the marginal returns to redundancy for all three of these scheduling policies. We explore the effect of shifting some jobs from a given class to a more redundant class. Under LRF, we show such a shift reduces the overall mean response time, but the improvement decreases as more jobs shift from the less redundant class to the more redundant class. That is, there is a *decreasing marginal benefit for redundancy*, indicating that the most significant response time improvements can be achieved with only a small amount of redundancy. While the result may seem unsurprising, the proof is surprisingly involved. We show via a coupling argument that increasing redundancy causes the sequence of departures to be earlier in the stochastic, sample-path sense. In the course of this proof, we derive an explicit stochastic expression for the overall decrease in response time, Δ, obtained when a single job is changed from nonredundant to redundant. We then use this expression to analyze the effect of a second switched job on Δ, yielding insight into second order effects. While our proof approach does not translate easily to PF scheduling, we observe empirically that the system behavior under PF is similar to that under LRF.

Surprisingly, under FCFS the same result does *not* hold: not only is there not necessarily a decreasing marginal benefit for increased redundancy, but *increased redundancy does not necessarily reduce response time*. This result, which is very counterintuitive when service times are i.i.d. and exponentially distributed, flies in the face of earlier results that suggest that more redundancy is always better [18, 17, 25]. We find that instead, the potential benefits of redundancy depend on the scheduling policy, and not only on the amount of redundancy.

We also study cross-derivative effects, that is, how the improvement of shifting jobs from class i to class j, where $S_i \subset S_j$, depends on the proportion of jobs that have shifted from some other class y to j, where $S_y \subset S_j$. Here we find that the improvement is *increasing* with more redundancy in *other* classes under all three policies. That is, there is an increasing marginal benefit for redundancy across different classes. This result is surprising in combination with the fact that there is not always an increasing marginal benefit for redundancy within the same class. And it has important implications for how best to configure redundancy systems when there is an opportunity to choose where to increase redundancy: mean response time is lower in a symmetric system than in an imbalanced system with the same fraction of redundant jobs.

The remainder of this paper is organized as follows. In Section 2 we review related work on redundancy systems and on other types of systems with flexibility. In Section 3 we define our system model. Section 4 presents our results on LRF scheduling, including both theoretical analysis and numerical examples illustrating our results. In Section 5 we study FCFS and PF scheduling, and in Section 6 we conclude.

2. Related Work

Redundancy has become an increasingly important area of study in recent years. Under the cancel-on-completion model considered in this paper, Koole and Righter [25] showed that under certain circumstances, the more redundancy the better; i.e., response time will be smallest if there is a single class of jobs that are replicated to all servers. This is consistent with Gardner et al.'s [17] closed-form results for mean response times in the symmetric Redundancy-d system, in which each job replicates to d randomly chosen servers. Our work differs from the above in that we assume that each job has a fixed class that indicates the set of servers to which it replicates. In the class-based setting, Gardner et al. [18] developed a closed form for the steady-state distribution on queue states, and Gardner et al. [16] found the steady-state response-time distributions for each class in nested redundancy systems, both under the assumption of Poisson arrivals and exponential service times.

Much of the existing work on redundancy systems assumes FCFS scheduling, and the question of how to schedule jobs in redundancy systems has begun to be addressed only recently. Sun et al. [30] investigate optimal scheduling policies in systems in which each job consists of multiple tasks, all of which must be completed. The tasks are allowed to be replicated at any server, unlike our class-based setting, and because they consider multi-task jobs much of the scheduling decision focuses on which of a job's tasks to schedule; hence their work does not apply to our setting. In the class-based setting, Bonald and Comte [12, 13] proposed and analyzed a balanced fairness scheduling policy under which response times are insensitive to the service time distribution. This notion of fairness is different from the type of fairness Gardner et al. [16] studied, in which the goal is for no class of jobs to be hurt by redundancy. Nageswaran and Scheller-Wolf [26] considered a similar concept of fairness, but their focus was on achieving fairness through dispatching (assuming FCFS scheduling), rather than on modifying the scheduling policy. See also [17, 19, 29, 31] for related analytical work for systems with redundancy and more general forms of data coding.

Note that, given exponential service times, our cancel-on-completion redundancy model is equivalent to a single-queue model in which more than one compatible server can work on a job at the same time. This is also known as "server collaboration" in the operations management literature. Similarly, the cancel-on-start redundancy model is equivalent to a single-queue system in which servers cannot collaborate, and to a system in which jobs are dispatched to servers immediately according to the join-the-shortest-work policy [9, 10]. The effects of server collaboration and of various scheduling policies in systems with server collaboration have been studied, for example, by Van Oyen, Gel, and Hopp [35] and Ahn and Righter [3]. Adan and Weiss [1] found the steady-state distribution on the queue and server states under FCFS for the noncollaborative version of our model (assuming exponential service times), and Adan et. al. [2] and Ayesta et al. [10] studied the relationship between the steady-state distributions for the collaborative and noncollaborative cases. The LRF policy, which minimizes overall response time in our collaborative (cancel-on-completion) model, has been shown to minimize response time in the noncollaborative case as well by Akgun et al. [5] (in this case, the policy is called Dedicated Customers First, DCF). Akgun et al. also argued, again for the noncollaborative case, that FCFS from a single queue results in only a small loss in overall response time relative to DCF, but is more fair across classes. In addition they showed that join-the-shortest-work routing is the most efficient routing policy, and that this corresponds to FCFS from a single queue.

Because increasing the set of servers that a job is replicated to adds flexibility to the system, our work fits into a broad stream of research showing diminishing marginal returns to flexibility. This has generally been examined in the (non-redundant) queueing context in terms of increasing the flexibility of servers, for example, through workforce cross training [6, 11, 22, 23, 32, 33]. The marginal effect of increasing customer flexibility has been studied for variants of join the shortest queue or join the shortest work. For example, Mitzenmacher [28] and Turner [34] have shown that "power of two" routing, i.e., choosing the shortest of two randomly selected queues for each job (or a subset of the jobs) is almost as good as routing to the shortest of all queues. See also Ayesta et al. [9] for an analysis of "power of d" or "redundancy-d" routing for the cancel-on-start (noncollaborative) case under FCFS. In work most closely related to ours, Akgun et al. [4] showed, for a symmetric two-server three-job-class system, that response times are decreasing and convex in the proportion of flexible jobs that can join the shortest queue; the coupling arguments that we use to reason about LRF in this paper follow a similar form to those presented by Akgun et al. He and Down [21] showed an asymptotic version of the same result. Diminishing marginal returns to flexibility for production networks and supply chain networks has been shown, for example, in [14, 20, 24].

3. Model

We consider a system with k servers and ℓ classes of jobs. Jobs arrive to the system with average rate λ; in some cases we assume that arrivals form a Poisson process. Each job is a class-i job, $1 \leq i \leq \ell$, independently with probability p_i, so class-i jobs arrive with average rate $\lambda_i = \lambda p_i$. Each class of jobs i is associated with a particular subset of the servers, $S_i = \{s|\text{server } s \text{ can serve class-}i \text{ jobs}\}$. Upon arrival, a class-$i$ job replicates itself by joining the queues at all servers in S_i.

A job's service time on server s is exponentially distributed with rate μ_s for all job classes. Service times are assumed to be independent across jobs and for the same job across multiple servers. A job is allowed to be in service at multiple servers simultaneously, in which case it is considered to be complete as soon as its first copy completes service. That is, if a job is in service on both servers s and r, its remaining time is distributed as $\min\{\text{Exp}(\mu_s), \text{Exp}(\mu_r)\} = \text{Exp}(\mu_s + \mu_r)$. When a job's first copy completes service, all remaining copies are cancelled immediately regardless of whether they are in the queue or in service at the other servers. Because the service times are exponential, this is equivalent to what is sometimes called "server collaboration" in the operations management literature, where servers can work together on a job with combined service rate equal to the sum of the servers' individual rates.

We consider a specific system structure called a *nested* system. In a nested system, for all classes of jobs i and j such that $i \neq j$, either (1) $S_i \subset S_j$, (2) $S_j \subset S_i$, or (3) $S_i \cap S_j = \emptyset$. Let $\mathcal{I}_i$ be the subsystem in which class-i is fully redundant. That is, the job classes in subsystem $\mathcal{I}_i$ are all classes j such that $S_j \subseteq S_i$, and the servers in subsystem $\mathcal{I}_i$ are the servers in S_i. Let $\mu_{\mathcal{I}_i}$ and $\lambda_{\mathcal{I}_i}$ denote the total service rate and total arrival rate respectively in subsystem $\mathcal{I}_i$. For stability, we assume that $\lambda_{\mathcal{I}_i} < \mu_{\mathcal{I}_i}$ for all classes i.

In much of this paper, for clarity we focus on a particular example of a nested system called the $\mathbb{W}$ model (see Figure 1). In the $\mathbb{W}$ model, there are two servers and three classes of jobs. Class-A jobs are non-redundant and join the queue at server 1 only. Class-B jobs are also non-redundant and join the queue at server 2 only. Class-R jobs are redundant and join the queues at both servers. We let p_A, p_B, and p_R denote the fraction of jobs that are class-A, class-B, and class-R respectively, where $p_A + p_B + p_R = 1$.

At this point the model is fully defined apart from the scheduling discipline. In the sections that follow, we consider three different policies: Least Redundant First (Section 4), First Come First Served (Section 5.1), and Primaries First (Section 5.2).

4. Convexity Under LRF

In this section we look at the effect of increasing the proportion of jobs that are redundant under LRF scheduling. We first show that as the proportion of redundant jobs increases, the departure process stochastically increases, so overall mean response time decreases (Lemma 1). We then show that the overall mean response time is convex in the proportion of redundant jobs (Theorem 1). That is, as more and more jobs become redundant, there is diminishing *marginal* benefit from an additional job becoming redundant.

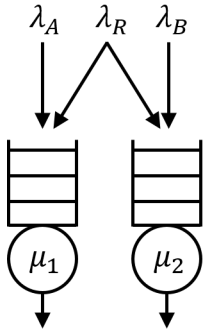


Figure 1: The $\mathbb{W}$ model. Class-A jobs arrive at rate $\lambda_A = \lambda p_A$ and join the queue at server 1 only, class-B jobs arrive at rate $\lambda_B = \lambda p_B$ and join the queue at server 2 only, and class-R jobs arrive at rate $\lambda_R = \lambda p_R$ and join the queues at both servers 1 and 2.

This is important because it tells us that *a little redundancy goes a long way*: we see the biggest response time gains from having just a small number of redundant jobs. In systems where there is a cost to redundancy, for example because redundant jobs' data must be replicated on multiple servers, this allows us to achieve the benefits of redundancy while not incurring high costs. This is analogous to results in systems without redundancy showing that a little flexibility goes a long way in reducing response time (see, e.g., [4, 28, 32], and see Section 2 for a more detailed discussion).

We begin by considering the specific case of the $\mathbb{W}$ model (see Section 3). In Section 4.3 we discuss how our arguments for the $\mathbb{W}$ model can be extended to general nested systems.

We prove convexity of mean response time as a function of p_R using a coupling argument. We first consider a fixed sample path, with a given p_R, and consider the effect of changing a single job from class-A (non-redundant) to class-R (redundant); this creates two coupled sample paths. In Lemma 1 we develop an explicit expression that captures the marginal benefit of switching one job from class-A to class-R. Corollaries 1 and 2 extend this result to allow any fraction of jobs to shift from class-A to class-R. We then consider the effect of changing additional jobs from class-A to class-R on the original marginal benefit (Theorem 1). This captures the second-order effect of increasing redundancy, and we show that the marginal benefit of class-A jobs becoming redundant decreases as more class-A jobs become redundant. We also consider the effect of changing a class-B job to a class-R job on the original marginal benefit of class-A jobs becoming redundant, and show that the marginal benefit of changing a class-A job to be redundant *increases* as more class-B jobs are redundant (Theorem 2).

4.1. First-Order Effects

We assume that the arrival processes and service rates are such that the first busy period ends (all servers become idle) at some finite random time. For Lemma 1 and Corollaries 1 and 2, we do not need any other conditions on the arrival process, except that it must be independent of the state of the system and of the scheduling policy. As stated in Section 3, we also require $\lambda_{\mathcal{I}_i} < \mu_{\mathcal{I}_i}$ for all classes i so that the system is stable.

We fix a sample path consisting of a given initial set of jobs, a given sequence of arrival times of jobs of each class (i.e., a sample path of three split Poisson processes with rates λ_i, $i = A, B, R$), and a given sequence of potential job completion times on each server (i.e., a sample path of two split Poisson processes with rates μ_1 and μ_2). We will couple two systems on this sample path; the difference between the two systems is that we shift some number of jobs from being class-A jobs in the first system to being class-R jobs in the second system. We call the shifted jobs "ϵ" jobs and denote the two systems $\text{Syst}(\epsilon_A)$, in which all of the ϵ jobs are class-A, and $\text{Syst}(\epsilon_R)$, in which all of the ϵ jobs are class-R. Because the LRF policy is nonidling (i.e., work-conserving), any differences in actual completion times between $\text{Syst}(\epsilon_A)$ and $\text{Syst}(\epsilon_R)$ must occur when a server idles in one system while in the other system it is busy.

Let $N^{\epsilon_i}(t)$ be the number of jobs in the ϵ_i system, $i = A, R$ We begin in Lemma 1 by assuming that there is a single ϵ job. Without loss of generality, define the arrival time of the ϵ job as time 0. Let Δ be the difference in the overall total response time between $\text{Syst}(\epsilon_A)$ and $\text{Syst}(\epsilon_R)$. We summarize this and other notation used in this section in Table 1. As part of our proof of Lemma 1 we derive an explicit expression for Δ, given in Equation (1), that will be needed later to prove convexity in Theorem 1.

Lemma 1. *On any sample path, if a single job is changed from class-A to class-R then $\Delta \geq 0$; indeed $\{N(t)^{\epsilon_A}\}_{t=0}^{\tau} \geq \{N^{\epsilon_R}(t)\}_{t=0}^{\tau}$ for all times τ.*

Proof. We will derive an expression for Δ, the difference in overall response time in $\text{Syst}(\epsilon_A)$ and $\text{Syst}(\epsilon_R)$. Differences in response time in the two systems could be experienced by jobs of any class, making the accounting challenging. However, we note that because service times are i.i.d. exponential and depend only on the server, the scheduling policy within a class has no effect on the mean response time for that class. Hence our first step is to modify our service discipline slightly for the ϵ job so that the entire difference in overall response time is experienced by the ϵ job (and all other jobs experience exactly the same response time in $\text{Syst}(\epsilon_A)$ and in $\text{Syst}(\epsilon_R)$).

Our modified scheduling policy is as follows. In accordance with LRF, among the regular (non-ϵ) jobs, class-A and class-B jobs have priority over class-R jobs. In $\text{Syst}(\epsilon_A)$, the ϵ job has lowest preemptive priority among class-A jobs. In $\text{Syst}(\epsilon_R)$, the ϵ job has highest preemptive priority among class-R jobs on server 1 and lowest preemptive priority among all jobs (both class-B and class-R) on server 2. That is, the ϵ job is

$N^{\epsilon_i}(t)$	The number of jobs in Syst(ϵ_i), $i = A, R$, at time t
Δ	The difference in total response time between Syst(ϵ_A) and Syst(ϵ_R)
X_1	The time at which the ϵ job will complete service at server 1 in Syst(ϵ_A), (for Thms 1 and 2, assuming there is no δ job)
X_2	The time at which the ϵ job will complete service at server 2 in Syst(ϵ_R), if server 1 did not exist (for Thms 1 and 2, assuming there is no δ job)
X_3	The time at which Syst(ϵ_A) first empties of all class-R jobs and the ϵ job (for Thms 1 and 2, assuming there is no δ job)
$\Delta(\delta_j)$	The difference in total response time when switching ϵ jobs from class-A to class-R, assuming δ jobs are class-j, $j = A, R$
X_1^δ	The time at which the δ job will complete service at server 1 in Syst(ϵ_i, δ_A), $i = A, R$, assuming there is no ϵ job
X_2^δ	The time at which the δ job will complete service at server 2 in Syst(ϵ_i, δ_R), $i = A, R$ if server 1 did not exist, assuming there is no ϵ job
X_3^δ	The time at which Syst(ϵ_i, δ_A), $i = A, R$ first empties of all class-R jobs and the δ job, assuming there is no ϵ job
$X_1(\delta_j)$	The time at which the ϵ job will complete service at server 1 in Syst(ϵ_A, δ_j), $j = A, R$, assuming that the δ job exists
$X_2(\delta_j)$	The time at which the ϵ job will complete service at server 2 in Syst(ϵ_R, δ_j), $j = A, R$ if server 1 did not exist, assuming that the δ job exists
$X_3(\delta_j)$	The time at which Syst(ϵ_A, δ_j), $j = A, R$ first empties of all class-R jobs and the ϵ job, assuming that the δ job exists
Z_1	The duration of a server-1 busy period started by the ϵ job and consisting only of class-A jobs and the ϵ job
Z_2	The duration of a server-2 busy period started by the ϵ job and consisting only of class-B jobs, class-R jobs, and the ϵ job
Z_3	The duration of a class-R busy period (i.e., the time until the system is next empty of class-R jobs), started by a single class-R job when server 1 is otherwise empty

Table 1: Summary of notation used for Lemma 1 (above double line) and Theorems 1 and 2.

treated the same way at server 1 in both systems (but still consistently with LRF), and if it is served by server 1 its effect on the waiting times of other jobs at that server is also the same. In Syst(ϵ_R), the copy of the ϵ job at server 2 is treated consistently with LRF, and it has no effect on the waiting times of other jobs at server 2.

Let X_1 denote the time at which the ϵ job will complete at server 1 in Syst(ϵ_A). Note that X_1 represents a remaining server-1 busy period consisting only of class-A jobs and the ϵ job. Let X_2 denote the time at which the ϵ job would complete at server 2 in Syst(ϵ_R), if server 1 did not exist (equivalently, if server 1 were busy working on class-A jobs for the entire time between the ϵ job's arrival and when the ϵ job completes at server 2). Note that X_2 represents a remaining server-2 busy period consisting of class-B and class-R jobs, including the ϵ job. In Syst(ϵ_R), the ϵ job completes at time $\min\{X_1, X_2\}$. We now consider both cases for when the ϵ job could complete in Syst(ϵ_R).

Case 1: $X_1 < X_2$**.** See Figure 2. Then at time X_1 both systems are empty of class-A jobs, and the ϵ job completes in both systems. At this point the two systems recouple: exactly the same set of jobs is present in both systems, and all jobs have the same completion times in both systems: $\Delta = 0$.

Case 2: $X_2 < X_1$**.** Then at time X_2 both systems are empty of class-B and class-R jobs, and the ϵ job departs from server 2 in Syst(ϵ_R) but remains in the queue at server 1 in Syst(ϵ_A), so starting at time X_2 Syst(ϵ_A) contains one more job than Syst(ϵ_R).

We now consider what happens at time X_1, the moment when the ϵ job departs in Syst(ϵ_A). Note that up until time X_1 the response times for all jobs besides the ϵ job continue to be the same in Syst(ϵ_A) and

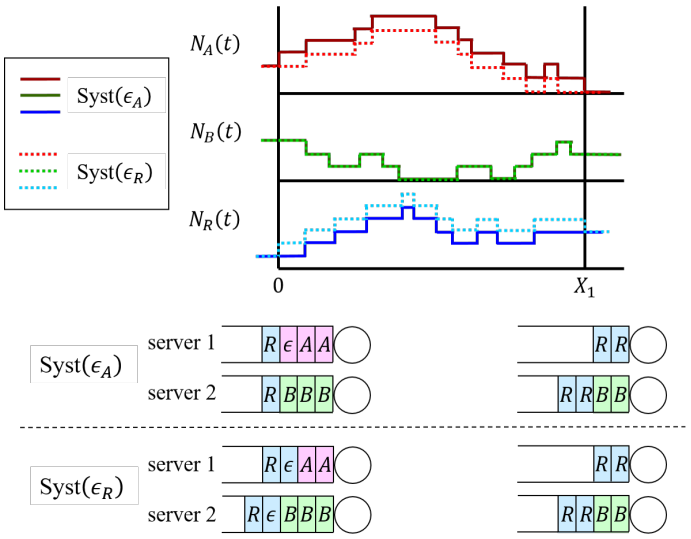


Figure 2: Lemma 1, case 1. The ϵ job completes at time X_1 in both systems.

Syst(ϵ_R).

Case 2.1: no class-R jobs are present in either system just before time X_1. See Figure 3. Then the ϵ job completes at time X_1 in Syst(ϵ_A) and the two systems recouple, so the ϵ job has no effect on any other jobs. Hence the difference in overall response time between the two systems is captured by the difference experienced by the ϵ job (namely the duration of time for which the ϵ job is present in Syst(ϵ_A) but not in Syst(ϵ_R)), which is $\Delta = X_1 - X_2$.

Case 2.2: there are class-R jobs present just before time X_1. See Figure 4. At time X_1 the ϵ job departs from Syst(ϵ_A), and a class-R job departs from Syst(ϵ_R). There continues to be one more job in Syst(ϵ_A) than in Syst(ϵ_R), where this extra job is a class-R job. We will now call this extra class-R job the ϵ job, and we will call the other class-R jobs, besides the ϵ job, regular jobs. We give the ϵ job lowest preemptive priority among all class-R jobs at both servers. Now we have the same number of regular class-R jobs in both systems, with the ϵ job having no effect on any other job. In Syst(ϵ_A), the ϵ job will complete at the end of the class-R busy period (the first time the system is empty of all class-R jobs, including the ϵ job), at which time the two systems recouple; call the moment at which this happens time X_3. In this case the ϵ job is in Syst(ϵ_A) but not Syst(ϵ_R) from time X_2 to time X_3, so $\Delta = X_3 - X_2$.

Putting everything together, we have that the difference in overall response time between Syst(ϵ_A) and Syst(ϵ_R), denoted by Δ, is completely captured by the difference in response time for the ϵ job, where

$$\begin{aligned}\Delta &= \begin{cases} 0, & X_1 < X_2 \\ X_1 - X_2, & X_1 > X_2 \text{ and no class-}R\text{ jobs present just before } X_1 \\ X_3 - X_2, & X_1 > X_2 \text{ and class-}R\text{ jobs present just before } X_1 \end{cases} \\ &= I_{X_2<X_1}(X_3 - X_2) \\ &\geq 0, \end{aligned} \tag{1}$$

where I is an indicator variable equal to 1 if the subscripted expression is true and 0 otherwise, and we define $X_3 = X_1$ if there are no class-R jobs in the system immediately before time X_1.

Since each class-A, class-B, and regular class-R job experiences the same response time in Syst(ϵ_A) as in Syst(ϵ_R), we thus have that there are at least as many jobs in Syst(ϵ_A) as in Syst(ϵ_R) at all times: $\{N^{\epsilon_A}(t)\}_{t=0}^{\tau} \geq \{N^{\epsilon_R}(t)\}_{t=0}^{\tau}$ for all τ. □

Lemma 1 tells us that on any sample path, switching a single job from class-A to class-R leads to a reduction in the number of jobs in the system, and hence in the overall system response time of all jobs in the sample path. Corollary 1 follows immediately from the repeated application of Lemma 1.

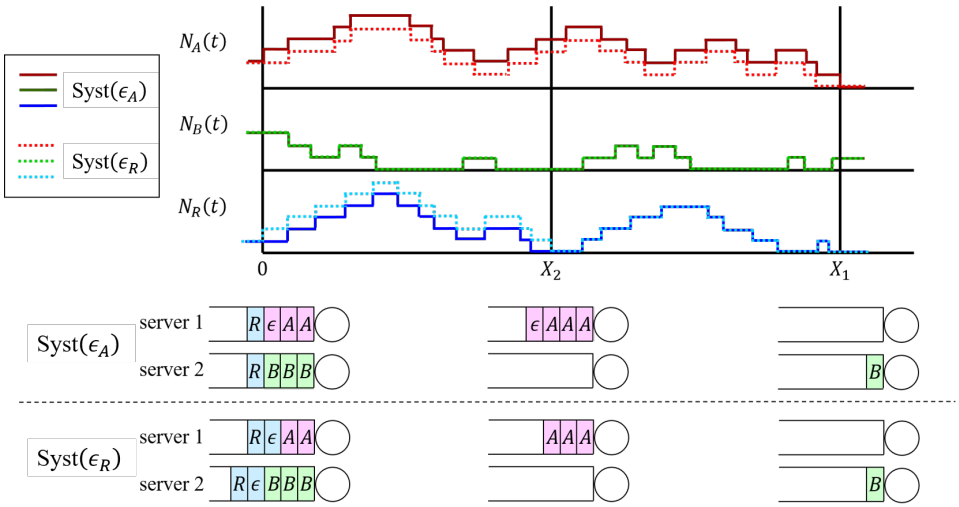


Figure 3: Lemma 1, case 2.1. The ϵ job completes at time X_2 in System II, and there are no class-R jobs in the system just before time X_1.

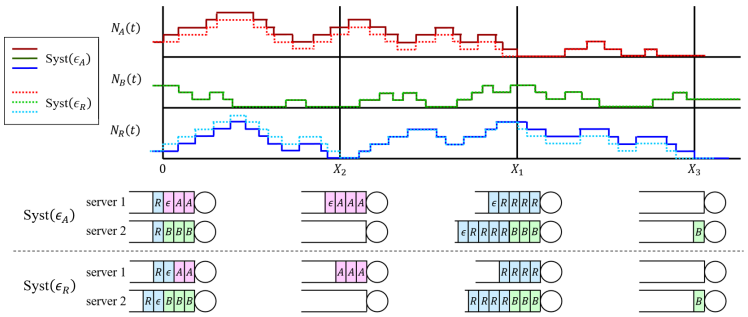


Figure 4: Lemma 1, case 2.2. The ϵ job completes at time X_2 in System II, and there are class-R jobs in the system just before time X_1.

Corollary 1. *On any sample path, if any number of jobs are changed from class-A to class-R, then* $\Delta \geq 0$ *and* $\{N^{\epsilon_A}(t)\}_{t=0}^{\tau} \geq \{N^{\epsilon_R}(t)\}_{t=0}^{\tau}$ *for all times* τ.

Finally, we consider response time in steady state as a function of the *proportion* of jobs that are redundant. Again suppose that we have a general exogenous arrival process, where the arrivals are independent of the state of the system and the scheduling policy, and such that the system is stable. Let p_A, p_B, and p_R denote the fraction of jobs that are class-A, class-B, and class-R respectively ($p_A + p_B + p_R = 1$). We assume that the class of each arriving job is chosen by i.i.d. splitting, so the jth arrival is of class i with probability p_i, $i = A, B, R$.

In our coupling, all arriving jobs have the same class in Syst(ϵ_A) and Syst(ϵ_R) except an ϵ fraction of the jobs, which are class-A in Syst(ϵ_A) and class-R in Syst(ϵ_R).

We define $T_i(p_R)$ as the steady state response time of class-i jobs and $T(p_R)$ as the overall steady state response time for all jobs when p_R is the fraction of jobs that are redundant. We assume p_B is held fixed, and p_A varies (inversely) with p_R, where $p_A = 1 - p_B - p_R$. Our stability conditions ensure that the steady state response times are well defined.

We remind the reader that for two random variables X and Y, we say $X \geq_{st} Y$, i.e., X is stochastically larger than Y, if $\mathbf{P}\{X > x\} \geq \mathbf{P}\{Y > x\}$ for all x. Equivalently, $X \geq_{st} Y$ if we can construct coupled versions $\tilde{X}$ and $\tilde{Y}$ (i.e., $\tilde{X}$ and $\tilde{Y}$ are on the same probability space) so that $\tilde{X} \geq \tilde{Y}$ with probability 1.

Corollary 2. *As the fraction of redundant jobs,* p_R*, increases, holding* p_B *constant:*

1. $T(p_R)$ *is stochastically decreasing, so* $\mathbb{E}[T(p_R)]$ *is decreasing.*
2. $T_A(p_R)$ *is stochastically decreasing, so* $\mathbb{E}[T_A(p_R)]$ *is decreasing.*
3. $T_B(p_R)$ *is constant, so* $\mathbb{E}[T_B(p_R)]$ *is constant.*

Proof. Using the above coupling between Syst(ϵ_A) and Syst(ϵ_R), part 1 follows immediately from Lemma 1. Parts 2 and 3 follow from the fact that class-A (respectively, class-B) jobs have preemptive priority over class-R jobs and therefore experience an independent G/M/1 queue consisting only of class-A (class-B) jobs. □

The mean response time for class-R jobs, $\mathbb{E}[T_R(p_R)]$, may either increase or decrease as p_R increases; we discuss this further in Section 4.4.

4.2. Second-Order Effects

We now turn to second-order effects of additional redundancy on overall mean response time. First we study convexity.

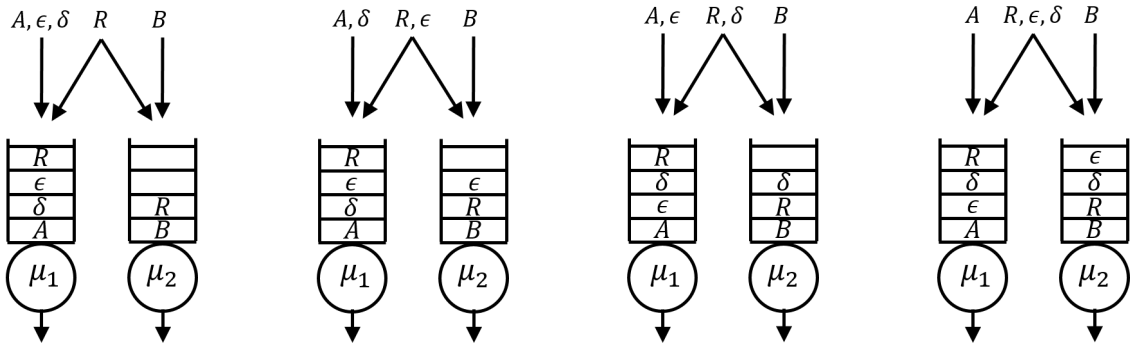
We again consider the effects of shifting some jobs, which we call ϵ jobs, from class-A to class-R. We will see how Δ, the difference in overall response time when the ϵ jobs are class-A versus class-R, changes if we increase p_R to $p_R + \delta$ while decreasing p_A to $p_A - \delta$. Later we consider the cross-derivative and determine how Δ changes if we increase p_R while decreasing p_B and holding p_A constant. Our proof of the convexity result requires Poisson arrivals, but the cross-derivative result does not. We conjecture that the convexity result holds more generally, e.g., for renewal arrival processes; while our proof does not easily extend to this setting, this conjecture is strongly supported by simulation results (see Section 4.4).

Theorem 1. *With Poisson arrivals, mean response time is convex in the fraction of redundant jobs.*

Proof. We construct four coupled systems with two types of tagged jobs, which we call ϵ jobs and δ jobs. In our coupling, all arriving jobs have the same class in all four systems, except the ϵ and δ jobs. In Syst(ϵ_i, δ_j) the ϵ jobs are class-i and the δ jobs are class-j, $i = A, R$, $j = A, R$.

The ϵ jobs capture the marginal effect on mean waiting time from increasing the number of redundant jobs starting from a fixed initial number (i.e., moving from Syst(ϵ_A, δ_A) to Syst(ϵ_R, δ_A) and moving from Syst(ϵ_A, δ_R) to Syst(ϵ_R, δ_R)). The δ jobs capture the change in the marginal effect of the ϵ jobs when starting with a larger number of redundant jobs (i.e., when starting in Syst(ϵ_A, δ_A) versus Syst(ϵ_A, δ_R)).

Our scheduling policy is as follows. In accordance with LRF, among the regular (non-ϵ and non-δ) jobs, class-A and class-B jobs have preemptive priority over class-R jobs. Both ϵ and δ jobs have lower priority



(a) Syst(ϵ_A, δ_A) (b) Syst(ϵ_R, δ_A) (c) Syst(ϵ_A, δ_R) (d) Syst(ϵ_R, δ_R)

Figure 5: The four systems that are coupled in the proof of Theorem 1. The four systems differ in their relative prioritizations of class-A, class-B, δ, and ϵ jobs.

than all class-A jobs and higher priority than all class-R jobs on server 1, and both have lowest priority of all jobs on server 2. When the δ job is class-A, it has higher priority than the ϵ job on server 1. When the δ job is class-R, it has lower priority than the ϵ job on server 1 and higher priority on server 2. Note that all of these priority orderings, illustrated in Figure 5, are consistent with LRF, and the ϵ jobs are treated the same as in the proof of Lemma 1.

Define $\Delta(\delta_A) = T^{(\epsilon_A,\delta_A)} - T^{(\epsilon_R,\delta_A)}$ to be the difference in response time when switching ϵ jobs from class-A to class-R when δ jobs are class-A (i.e., when going from Syst(ϵ_A, δ_A) to Syst(ϵ_R, δ_A), and define $\Delta(\delta_R) = T^{(\epsilon_A,\delta_R)} - T^{(\epsilon_R,\delta_R)}$ to be the difference in response time when switching ϵ jobs from class-A to class-R when δ jobs are class-R (i.e., when going from Syst(ϵ_A, δ_R) to Syst(ϵ_R, δ_R). We will show that $\mathbb{E}[\Delta(\delta_A)] \geq \mathbb{E}[\Delta(\delta_R)]$, where the expectation is taken over all sample paths.

As in our proof of Lemma 1, we will begin by assuming that the system contains a single ϵ job and a single δ job (Lemma 2). Theorem 1 then will follow immediately from Corollary 3, in which we allow the system to contain any fixed number of ϵ and δ jobs, and Corollary 4, in which we allow a fraction of jobs to be ϵ jobs and δ jobs.

Lemma 2. *If the system contains a single ϵ job and a single δ job, then $\mathbb{E}[\Delta(\delta_A)] \geq \mathbb{E}[\Delta(\delta_R)]$.*

Proof. We will assume for now that the ϵ job arrives before the δ job. As in the proof of Lemma 1, when the departure of the ϵ job results in Syst(ϵ_A, δ_A) and Syst(ϵ_A, δ_R) having one extra class-R job relative to Syst(ϵ_R, δ_A) and Syst(ϵ_R, δ_R), we will "relabel" a regular class-R job to be called the ϵ job, and give this relabeled job lowest preemptive priority among all class-R jobs. Let time 0 be the arrival time of the ϵ job and let $\tau > 0$ be the arrival time of the δ job. We define X_1, X_2, and X_3 as in the proof of Theorem 1 (in these definitions we assume that the δ job does not exist). Define X_1^δ, X_2^δ, and X_3^δ analogously for the δ job, assuming that the ϵ job does not exist.

Let $X_1 + Z_1$ be the time at which the ϵ job departs from server 1 in Syst(ϵ_A, δ_A), assuming $0 \leq \tau \leq X_1$. X_1 represents the duration of a server-1 busy period started by the class-A work already present in the queue when the ϵ job arrives, and consisting only of class-A jobs and the δ job. Z_1 represents the duration of a server-1 busy period started by the ϵ job and consisting only of class-A jobs and the ϵ job. Similarly, let $X_2 + Z_2$ be the time at which the ϵ job would depart from server 2 in Syst(ϵ_R, δ_R), assuming $\tau \leq X_2$. X_2 represents the duration of a server-2 busy period started by the class-B and class-R work already present in the queue when the ϵ job arrives, and consisting only of class-B jobs, class-R jobs, and the δ job. Z_2 represents the duration of a server-2 busy period started by the ϵ job and consisting of class-B jobs, class-R jobs, and the ϵ job. Let $X_3 + Z_3$ be the time until there are no class-R jobs in the system if we add one more class-R job at time X_3, i.e., Z_3 represents the duration of a class-R busy period started by a single class-R job when server 1 is otherwise empty. Note that this class-R busy period could end with a service completion at either server 1 or server 2.

Finally, we define $X_i(\delta_j)$ analogously, where we now assume that the δ job does exist and δ_j denotes the class of the δ job, $j = A, R$. For example, $X_1(\delta_A)$ represents the time at which the ϵ job would complete at

server 1 if the δ job is a class-A job; this is the time at which the ϵ label is reassigned to a regular class-R job in Syst(ϵ_A, δ_A).

We can immediately write the following expressions for the $X_i(\delta_j)$ terms for $i = 1, 2$, $j = A, R$:

$$\begin{aligned} X_1(\delta_A) &= X_1 + I_{\tau \le X_1} \cdot Z_1 \\ X_1(\delta_R) &= X_1 \\ X_2(\delta_A) &= X_2 \\ X_2(\delta_R) &= X_2 + I_{\tau \le X_2} \cdot Z_2. \end{aligned}$$

We will derive expressions for $X_3(\delta_j)$, $j = A, B$, below.

By the argument used to derive equation (1) in Lemma 1, we have the following for $i = A, R$:

$$\Delta(\delta_i) = I_{X_2(\delta_i) < X_1(\delta_i)}(X_3(\delta_i) - X_2(\delta_i)). \tag{2}$$

If $X_2 > X_1$, then $\Delta(\delta_R) = 0 \le \Delta(\delta_A)$ and we are done. Similarly, if $\tau > \max\{X_2, X_3\}$ then the δ job arrives after the ϵ job has already departed, so $\Delta(\delta_R) = \Delta(\delta_A)$, and again we are done. We now consider the case in which $X_2 < X_1 \le X_3$ and $\tau \le X_3$. The proof of Lemma 2 will follow from the following lemmas.

Lemmas 3-5 address the case in which $\tau < X_1$, meaning that the δ job has arrived before the ϵ job departs in Syst(ϵ_A, δ_A) and Syst(ϵ_A, δ_R). We begin in Lemma 3 by studying Syst(ϵ_A, δ_A) and Syst(ϵ_R, δ_A), in which the δ job is class-A.

Lemma 3. *If $\tau < X_1$ and $X_2 < X_1$, then $\Delta(\delta_A) \ge X_3 + Z_3 - X_2$.*

Proof. We know that $X_2(\delta_A) = X_2$, so in Syst(ϵ_R, δ_A) the (class-R) ϵ job departs at time X_2.

All that remains is to show that $X_3(\delta_A) \ge X_3 + Z_3$; $X_3(\delta_A)$ is the time at which the ϵ job will depart from Syst(ϵ_A, δ_A). If the (class-A) δ job did not exist, then the (class-A) ϵ job would depart the system at time X_1. If there were no class-R jobs present at X_1, then $X_3 = X_1$ by definition. If there were class-R jobs present at time X_1, then the ϵ label would be reassigned to the lowest priority class-R job, which would leave the system at time $X_3 > X_1$. Here X_3 represents the time at which all class-R jobs in the system would have completed if there were no δ job. Instead, because $\tau \le X_1$, the δ job departs at time X_1 instead of the ϵ job. At time $X_1 + Z_1$ the ϵ job departs and, if there are class-R jobs present, the ϵ label is reassigned to the lowest priority class-R job.

We know that $X_3(\delta_A) \ge X_1(\delta_A) = X_1 + Z_1$, so if $X_3 + Z_3 < X_1 + Z_1$ we are done. Suppose $X_3 + Z_3 \ge X_1 + Z_1$. If there are no regular class-R jobs in the system at $X_1 + Z_1$, then $X_1 + Z_1 = X_3 + Z_3 = X_3(\delta_A)$, and we are done. If there are regular class-R jobs in the system at time $X_1 + Z_1$, then the ϵ job becomes the lowest priority class-R job and leaves at time $X_3(\delta_A) = X_3 + Z_3$. Hence $X_3(\delta_A) = \max\{X_1 + Z_1, X_3 + Z_3\} \ge X_3 + Z_3$ as desired. □

Lemmas 4 and 5 deal with Syst(ϵ_A, δ_R) and Syst(ϵ_R, δ_R), in which the δ job is class-R.

Lemma 4. *If $\tau < X_1$ and $X_2 < X_1$ and $X_2^\delta < X_1^\delta$, then $\Delta(\delta_R) \le X_3 - X_2$.*

Proof. In this case, the (class-R) δ job has already completed before time $X_1^\delta = X_1 = X_1(\delta_R)$, so $X_3(\delta_R) = X_3$, and $\Delta(\delta_R) = X_3 - (X_2 + I_{\tau \le X_2} \cdot Z_2)$ and we are done. □

Lemma 5. *If $\tau < X_1$ and $X_2 < X_1$ and $X_2^\delta \ge X_1^\delta$, then $\Delta(\delta_R) = X_3 + Z_3 - X_2$.*

Proof. In this case, the (class-R) δ job completes at time X_1, and the ϵ label is reassigned to the lowest priority class-R job, which completes at time $X_3 + Z_3$. □

Finally, Lemmas 6 and 7 handle the case in which $X_1 < \tau$, meaning that when the δ job arrives, the ϵ job has departed from all four systems, and Syst(ϵ_A, δ_A) and Syst(ϵ_A, δ_R) contain an extra "relabeled" ϵ job.

Lemma 6. *If $X_2 < X_1 \le \tau < X_3$ and $X_2^\delta < X_1^\delta$, then $\Delta(\delta_A) = X_3 - X_2$ and $\Delta(\delta_R) = X_3 + Z_3 - X_2 = \Delta(\delta_A) + Z_3$.*

Proof. In Syst(ϵ_R, δ_A) and Syst(ϵ_R, δ_r), the ϵ job completes at time X_2, before the δ job arrives. In Syst(ϵ_A, δ_R), the ϵ label is reassigned to the lowest priority class-R job at time X_1. In Syst(ϵ_A, δ_A), the ϵ job leaves at time X_3. In Syst(ϵ_A, δ_R), the δ job leaves at time $X_2^\delta = X_3$ and the ϵ job leaves at time $X_3 + Z_3$. □

Lemma 7. *If $X_2 < X_1 \leq \tau < X_3$ and $X_2^\delta \geq X_1^\delta$, then $\Delta(\delta_A) = \Delta(\delta_R) = X_3 + Z_3 - X_2$.*

Proof. Since $X_1^\delta \leq X_2^\delta$, the δ job completes at $X_1^\delta \leq X_3$ in all systems. Note that a class-R job would have completed at X_1^δ if there were no δ job. Therefore, starting from time X_1^δ, there is one more class-R job in all systems than there would have been if the δ job had not arrived, so $X_3(\delta_A) = X_3(\delta_R) = X_3 + Z_3$. □

Putting Lemmas 3-7 together, we have $\Delta(\delta_R) \leq \Delta(\delta_A)$ in all cases except, from Lemma 6, when $X_2 < X_1 \leq \tau < X_3$ and $X_2^\delta < X_1^\delta$. In this case $\Delta(\delta_R) = X_3 + Z_3 - X_2 = \Delta(\delta_A) + Z_3$. From Lemmas 3 and 4, we have $\Delta(\delta_R) \leq \Delta(\delta_A) - Z_3$ when $\tau \leq X_2 < X_1$. Therefore, we will have $\mathbb{E}[\Delta(\delta_R)] \leq \mathbb{E}[\Delta(\delta_A)]$ as long as

$$\Pr\{\tau \leq X_2 < X_1 | X_2 < X_1 < X_3\} \geq \Pr\{X_1 < \tau < X_3 | X_2 < X_1 < X_3\}. \tag{3}$$

Here is the first time we use our assumption of Poisson arrivals. Let X_R be the time, starting in steady state, until the first moment at which there are no class-R jobs. Recall that X_2 represents a busy period for server 2 started in steady state (from PASTA), consisting of class-B and class-R jobs at server 2 and assuming no class-R jobs are served on server 1, and X_1 represents a busy period for class-A jobs in steady state. Therefore $[X_R | X_2 \leq X_1 < X_3] =_{st} [X_2 | X_2 < X_1 < X_3]$. On the other hand, $[X_3 - X_1 | X_2 \leq X_1 < X_3]$ represents a remaining busy period for class-R arrivals only, to either server, starting at time X_1, and at X_1 we know that earlier in the busy period, at time X_2, there were no class-R jobs. Therefore

$$[X_3 - X_1 | X_2 < X_1 < X_3] \leq_{st} [X_R | X_2 < X_1 < X_3], \tag{4}$$

and the result follows. The case in which the δ job arrives first is similar, so is omitted. □

This completes the proof of Lemma 2. □

By repeatedly applying Lemma 2 to allow the system to contain additional ϵ and δ jobs, we immediately obtain Corollary 3.

Corollary 3. *For a fixed number of ϵ jobs and a fixed number of δ jobs, $\mathbb{E}[\Delta(\delta_A)] \geq \mathbb{E}[\Delta(\delta_R)]$.*

Our last step is to define our four systems so that ϵ and δ *fractions* of jobs shift from being class-A to class-R, rather than a fixed number of jobs. We now have (holding p_B constant):

- In Syst(ϵ_A, δ_A), p_A fraction of jobs are class-A and p_R fraction of jobs are class-R.
- In Syst(ϵ_A, δ_R), $p_A - \delta$ fraction of jobs are class-A and $p_R + \delta$ fraction of jobs are class-R.
- In Syst(ϵ_R, δ_A), $p_A - \epsilon$ fraction of jobs are class-A and $p_R + \epsilon$ fraction of jobs are class-R.
- In Syst(ϵ_R, δ_R), $p_A - \epsilon - \delta$ fraction of jobs are class-A and $p_R + \epsilon + \delta$ fraction of jobs are class-R.

Using this definition, Corollary 4 immediately follows.

Corollary 4. *If some fraction of the jobs in the system are ϵ jobs and some fraction are δ jobs, then $\mathbb{E}[\Delta(\delta_A)] \geq \mathbb{E}[\Delta(\delta_R)]$.*

Theorem 1 follows directly from Corollary 4.

Next we consider the effect of changing both class-A and class-B jobs to become redundant class-R jobs. We find that as more jobs shift from class-B to class-R, the marginal benefit of shifting jobs from class-A to class-R *increases*. This is surprising given our earlier result (Theorem 1) that the marginal benefit decreases as more jobs shift from class-A to class-R while holding the fraction of class-B jobs constant. Theorem 2 tells us that we can achieve a significant response time benefit from a *symmetric* system assuming we allow some class-B jobs to become redundant whenever we allow some class-A jobs to become redundant. Here we no longer require the arrival process to be Poisson; it can be any general exogenous process, where the arrivals

must be independent of the state of the system and of the scheduling policy. We assume that the class of each arriving job is chosen by i.i.d. splitting, so the jth arrival is of class i with probability p_i, $i = A, B, R$. Let $T(p_A, p_B)$ denote the response time in a system in which p_A fraction of the jobs are class-A, p_B fraction of the jobs are class-B, and $p_R = 1 - p_A - p_B$ fraction of the jobs are class-R.

Theorem 2. *For a general exogenous arrival process and an arbitrary system busy period, we can define the systems on the same probability space so that*

$$T(p_A, p_B) - T(p_A - \epsilon, p_B) < T(p_A, p_B - \delta) - T(p_A - \epsilon, p_B - \delta) \tag{5}$$

with probability 1. That is, as an increasing fraction of jobs shift from class-B to class-R, the marginal benefit of shifting jobs from class-A to class-R increases.

Proof. Our setup is the same as before, with the ϵ job as defined in the proof of Theorem 1 so that it captures the marginal benefit of increasing the fraction of class-R jobs while decreasing the fraction of class-A jobs. The δ job can either be a class-B job or a class-R job. We couple four systems, where in $\text{Syst}(\epsilon_i, \delta_j)$ the ϵ job is class-i, $i = A, R$ and the δ job is class-j, $j = B, R$.

As in the proof of Theorem 1, let $T^{(\epsilon_i, \delta_j)}$ denote the response time in $\text{Syst}(\epsilon_i, \delta_j)$, and let $\Delta(\delta_j) = T^{(\epsilon_A, \delta_j)} - T^{(\epsilon_R, \delta_j)}$, $j = B, R$.

Again we begin by assuming that the system contains a single ϵ job and a single δ job. Using the same notation as in the proof of Theorem 1, and assuming the ϵ job arrives at time 0 and the δ job arrives at time $\tau > 0$, we have

$$\begin{aligned} X_1(\delta_B) &= X_1(\delta_R) = X_1 \\ X_2(\delta_B) &= X_2(\delta_R) = X_2 + I_{\tau \le X_2} \cdot Z_2. \end{aligned}$$

We also have $X_3(\delta_B) = X_3(\delta_R)$ if $\tau \le X_1$ and $X_2 + I_{\tau \le X_2} \cdot Z_2 < X_1$. Hence $\Delta(\delta_B) = \Delta(\delta_R)$, except in the case in which $X_2 < X_1 < \tau \le X_3$. In this case the ϵ job is no longer in the system at time τ in $\text{Syst}(\epsilon_R, \delta_B)$ and $\text{Syst}(\epsilon_R, \delta_R)$, while in $\text{Syst}(\epsilon_A, \delta_B)$ and $\text{Syst}(\epsilon_A, \delta_R)$ the ϵ label has been reassigned to the lowest priority class-R job. If in $\text{Syst}(\epsilon_R, \delta_R)$ the δ job is served by server 2, then in $\text{Syst}(\epsilon_R, \delta_B)$ the δ job is served at the same time on server 2, so again $\Delta(\delta_B) = \Delta(\delta_R)$.

Finally, if in $\text{Syst}(\epsilon_R, \delta_R)$ the δ job is served by server 1, then it is served at time X_3, and $X_3(\delta_R) = X_3 + Z_3$ while $X_3(\delta_B) = X_3 \le X_3(\delta_R)$. In this case $\Delta(\delta_B) < \Delta(\delta_R)$.

By repeatedly applying this argument, we have that if a fixed number of δ jobs shifts from class-B to class-R, the marginal benefit of shifting a fixed number of ϵ jobs from class-A to class-R increases.

The theorem immediately follows from this result. □

Corollary 5. *If $\mu_1 = \mu_2$ and p_R is held constant, then $\mathbb{E}[T]$ is minimized when $p_A = p_B$.*

Proof. By Theorem 1,

$$\mathbb{E}[T(p + x, p - x)] - \mathbb{E}[T(p, p - x)] > \mathbb{E}[T(p, p - x)] - \mathbb{E}[T(p - x, p - x)]. \tag{6}$$

By Theorem 2,

$$\mathbb{E}[T(p, p - x)] - \mathbb{E}[T(p - x, p - x)] > \mathbb{E}[T(p, p)] - \mathbb{E}[T(p - x, p)]. \tag{7}$$

Since $\mu_1 = \mu_2$, we also have

$$\mathbb{E}[T(p - x, p)] = \mathbb{E}[T(p, p - x)]. \tag{8}$$

Combining (6), (7), and (8), we have

$$\mathbb{E}[T(p + x, p - x)] - \mathbb{E}[T(p, p - x)] > \mathbb{E}[T(p, p)] - \mathbb{E}[T(p, p - x)],$$

and so

$$\mathbb{E}[T(p + x, p - x)] > \mathbb{E}[T(p, p)].$$

□

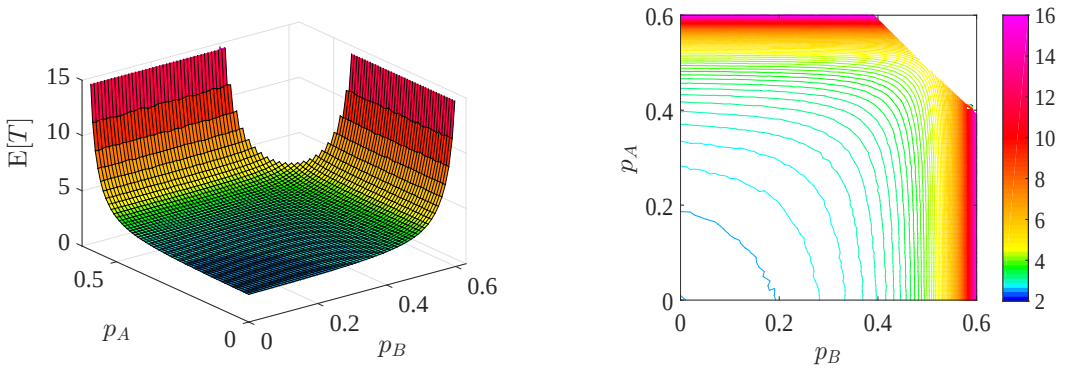


Figure 6: Mean response time under LRF as a function of p_A and p_B when $\lambda = 1.6$ and $\mu_1 = \mu_2 = 1$.

4.3. Generalizing to Larger Nested Systems

While the proofs presented in the preceding sections apply specifically to the $\mathbb{W}$ model, our approach extends to any nested system. Suppose we have a nested redundancy system, and let class A be some class that shares servers with a more-redundant class. Let class R be the class with smallest $|S_R|$ such that $S_A \subset S_R$. To prove the analogue of Lemma 1 for general nested systems, we need only modify very slightly the definitions used in the proof of Lemma 1.

Lemma 8. *In a general nested system with two classes A and R such that $S_A \subset S_R$, mean response time is convex in p_R, holding $p_A + p_R$ constant and holding p_i constant for all classes $i \neq A, R$.*

Proof. (Sketch). We again consider a fixed sample path. We couple two systems, where we switch one job, called the ϵ job, from being a class-A job in Syst(ϵ_A) to being a class-R job in Syst(ϵ_R). Unlike in the $\mathbb{W}$ model, S_A may now consist of more than one server, so we need to specify in slightly more detail how we prioritize the ϵ job. In Syst(ϵ_A) we let the ϵ job have lowest priority among class A jobs, and in Syst(ϵ_R) we let it have highest priority among class-R jobs on the servers in S_A and lowest priority among class-R jobs on the servers in $S_B = S_R \backslash S_A$. In addition, there may be other classes besides class A and class R that share the servers in S_A; such a class i may have $S_i \subset S_A$ or $S_i \supset S_R$. We therefore need to redefine X_1 and X_2, which in Lemma 1 referred to class-A and class-R busy periods. We redefine X_1 as the remaining class-A busy period under LRF in Syst(ϵ_A), where here by a "class-A busy period" we mean the time until the system is empty of class-A *and higher priority* jobs, and the ϵ job, on the servers in S_A. We redefine X_2 as a remaining class-R busy period for the servers in S_B, assuming the servers in S_A do not exist; that is, a "class-R busy period" is the time until the servers in S_B are empty of class-R jobs. Given these redefinitions, the rest of the argument follows analogously with the proof of Lemma 1 for the $\mathbb{W}$ model. □

Similarly, redefining X_i^δ and $X_i(\delta_j)$ to account for the additional servers and job classes allows us to extend the results of Theorems 1 and 2 to general nested systems.

4.4. Numerical Results

In this section we use simulation to illustrate the analytical results presented above. Unless otherwise specified, the results shown are for the $\mathbb{W}$ model (see Figure 1).

4.4.1. Overall Mean Response Time

Figure 6 shows mean response time under LRF as a function of p_A and p_B ($p_R = 1 - p_A - p_B$) when $\mu_1 = \mu_2 = 1$. The monotonicity of mean response time and the convex shape are clear from the left-hand figure: holding p_B constant, as p_A increases mean response time increases convexly, consistent with Theorem 1. The right-hand figure shows a contour map of mean response time as a function of p_A and p_B; warmer colors represent higher mean response time, while cooler colors represent lower mean response time. The contour map illustrates the cross-derivative result (Theorem 2). When p_B is high, the contour lines

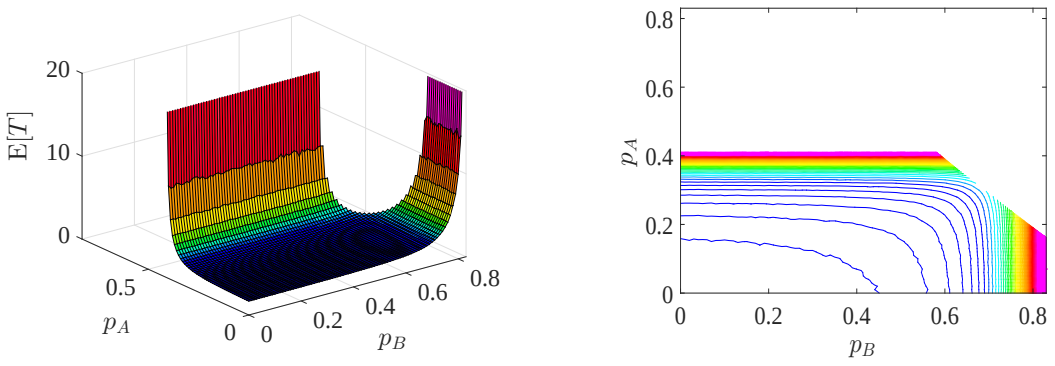
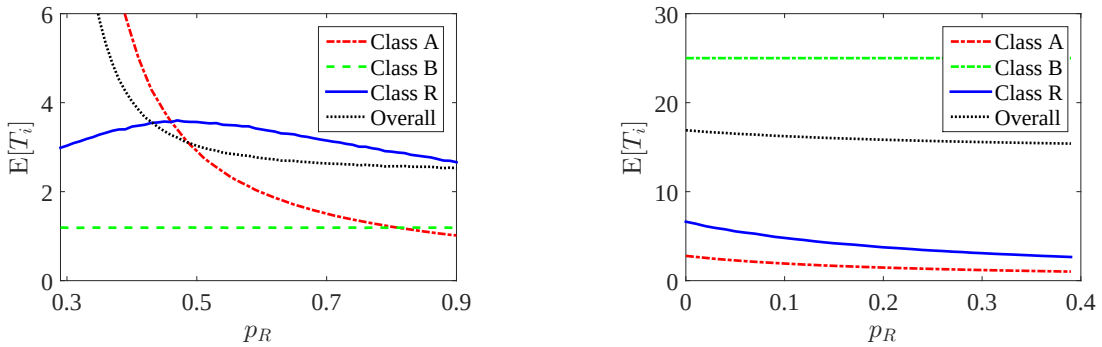


Figure 7: Mean response time under LRF as a function of p_A and p_B when $\lambda = 2.4$, $\mu_1 = 1$, and $\mu_2 = 2$.

are nearly vertical, indicating that changing p_A has very little effect on mean response time. But when p_B is low, the contour lines shift to being nearly horizontal, indicating that decreasing p_A significantly reduces mean response time. Both results are symmetric in p_A and p_B, indicating that for a fixed p_R it is best to set $p_A = p_B$ (Theorem 5).

Figure 7 shows mean response time as a function of p_A and p_B in an asymmetric system in which server 2 has twice the speed of server 1. When the servers no longer have equal rates, mean response time no longer is symmetric in p_A and p_B, but the monotonicity and convexity results still hold.

4.4.2. Per-Class Mean Response Time



(a) $p_B = 0.1$ (b) $p_B = 0.6$

Figure 8: Per-class mean response times under LRF as a function of p_R when (a) $p_B = 0.1$ and (b) $p_B = 0.6$. Here $\mu_1 = \mu_2 = 1$, $\lambda = 1.6$, and p_A decreases as p_R increases.

Corollary 2 tells us that as the fraction of redundant jobs increases (while holding p_B constant), the mean response time for class-A jobs is decreasing, the mean response time for class-B jobs is constant, and the overall mean response time is decreasing. Indeed, our simulations corroborate this result (see Figure 8). On the other hand, our analytical results do not tell us anything about the effect of p_R on the mean response time for class-R jobs. We see in Figure 8(a) that when $p_B = 0.1$ the class-R mean response time is *concave* and *non-monotonic* in p_R. At first, increasing p_R slightly results in an increase in class-R's mean response time. However, after reaching a maximum at around $p_R = 0.5$, the mean response time then *decreases*. This is because when p_R is very low (i.e., p_A is high), nearly all class-R jobs receive service at server 2, where they wait behind a relatively small number of class-B jobs. As p_R increases slightly from this low starting point (i.e., class-A jobs switch to being class-R), the traffic at server 2 increases slightly and class-R jobs experience longer mean response times. Once p_R is high enough some class-R jobs end up in service on server

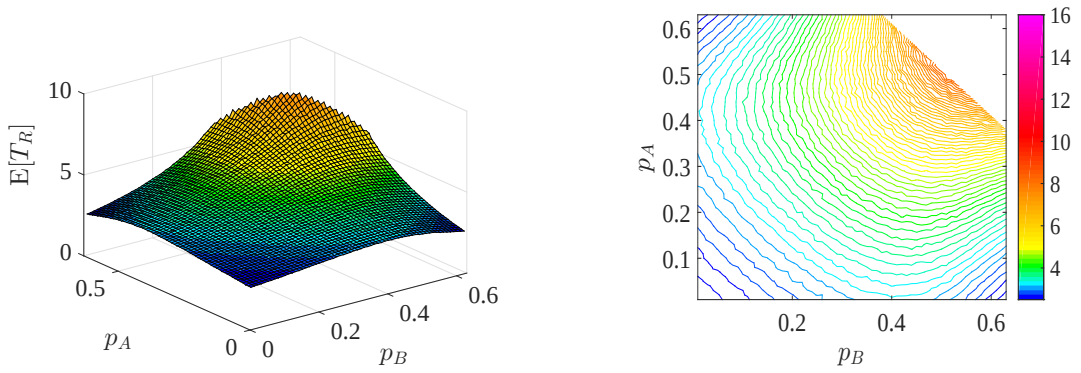
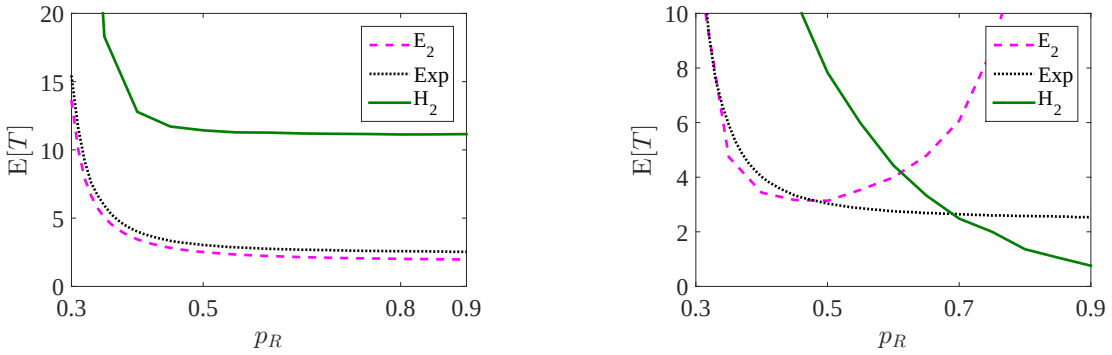


Figure 9: Mean response time for class-R jobs as a function of p_A and p_B under LRF when $\mu_1 = \mu_2 = 1$ and $\lambda = 1.6$.



(a) General interarrival times

(b) General service times

Figure 10: Mean response time under LRF where the mean arrival rate is $\lambda = 1.6$ and the mean service rate at each server is $\mu_1 = \mu_2 = 1$. Here $p_B = 0.1$ is constant and p_A varies inversely with p_R.

1, at which point the benefit of getting to experience the shorter waiting time across two servers begins to outweigh the cost of increasing the traffic at one of those two servers.

When the fraction of jobs that are class-B is higher ($p_B = 0.6$, Figure 8(b)), the shape of the class-R mean response time changes. Now, the mean response time for class-R jobs is *convex and decreasing* as p_R increases. When p_B is high and λ is high, the load on server 2 due to class-B jobs is high so only a small fraction of class-R jobs receive service at server 2. Increasing p_R slightly increases the number of jobs that benefit from waiting in both queues, so the class-R mean response time decreases. As p_R increases the marginal benefit of further jobs becoming redundant decreases—the class-R mean response time is convex—because with high load at server 2 fewer and fewer "new" class-R jobs actually end up in service on server 2. Figure 9 illustrates the effect of changing p_A and p_B on the class-R mean response time in more detail.

4.4.3. Relaxing Exponentiality Assumptions

In this section we study how relaxing our assumptions of Poisson arrivals and exponential service times affects mean response time. Our goal is to understand if our results hold more generally, or if they require exponentiality assumptions. Figure 10(a) shows mean response time under LRF in a system with general (non-Poisson) arrivals, where $p_B = 0.1$ is fixed. Our monotonicity result in Lemma 1 holds for any exogenous arrival process, and indeed we see that mean response time is monotonically decreasing in p_R with both Erlang and Hyperexponential interarrival times. As expected, when interarrival times are less variable mean response time decreases relative to Poisson arrivals, whereas when interarrival times are more variable mean response time increases.

Our proof of Theorem 1, which states that mean response time is convex in p_R, requires Poisson arrivals,

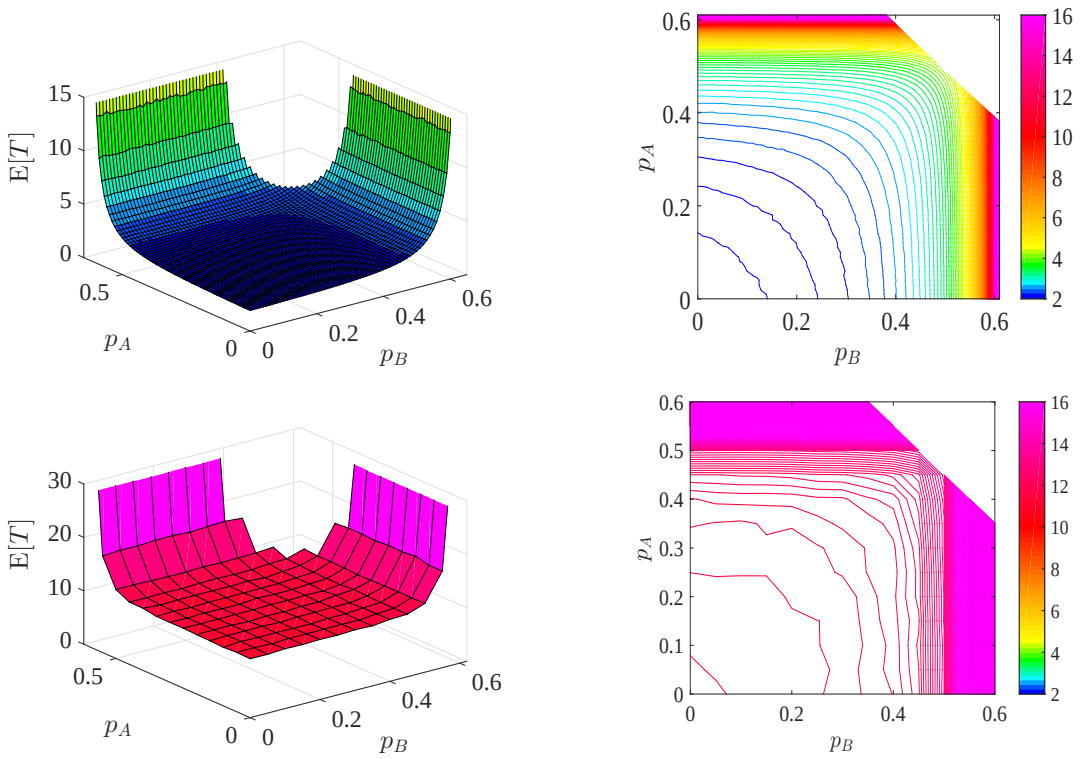


Figure 11: Mean response time under LRF where $\mu_1 = \mu_2 = 1$ and the interarrival times follow a two-phase Erlang distribution with mean rate 1.6 (top row) and a two-phase Hyperexponential distribution with mean rate 1.6 and squared coefficient of variation 10 (bottom row).

though Corollary 2, that mean response time is decreasing, does not. Our numerical results suggest that the convexity result holds more generally for any exogenous arrival process.

Conjecture 1. *Under LRF, mean response time is convex in p_R for any exogenous arrival process.*

Figure 11 supports this conjecture: with both Erlang and Hyperexponential interarrival times, overall mean response time under LRF appears to be convex. The contour plots (right-hand column of Figure 11) show the cross-derivative result from Theorem 2 (which holds for general interarrival times): as p_B increases, there is decreasing marginal benefit from further class-A jobs becoming redundant.

In contrast, whether our monotonicity and convexity results hold under general *service* times appears to depend on the particular characteristics of the service time distribution. Figure 10(b) shows mean response time under LRF with general service times and Poisson arrivals, where again $p_B = 0.1$ is constant. Under Hyperexponential service times mean response time appears to be both monotonically decreasing and convex in p_R. Perhaps counterintuitively, when p_R is high mean response time actually can be lower than with less-variable exponential service times. This happens because when service times are i.i.d. and more highly variable, there is a larger potential gain from running a job on multiple servers; when p_R is low the negative effects of the non-redundant jobs having highly variable service times begin to dominate and mean response time becomes very high.

On the other hand, under Erlang service times mean response time is not monotonically decreasing in p_R. When p_R is very high the system can even become unstable. This is because a job that draws two i.i.d. Erlang service times likely sees two service times that are reasonably close together. The consequence is that redundancy adds load to the system, causing instability when the arrival rate is sufficiently high. As p_R decreases slightly fewer jobs actually run on both servers, so less work is wasted, and instead the redundant jobs get to benefit from experiencing the shorter of two queueing times. When p_R decreases further, this queueing benefit is lost and mean response time again increases. Figure 12 shows mean response time with

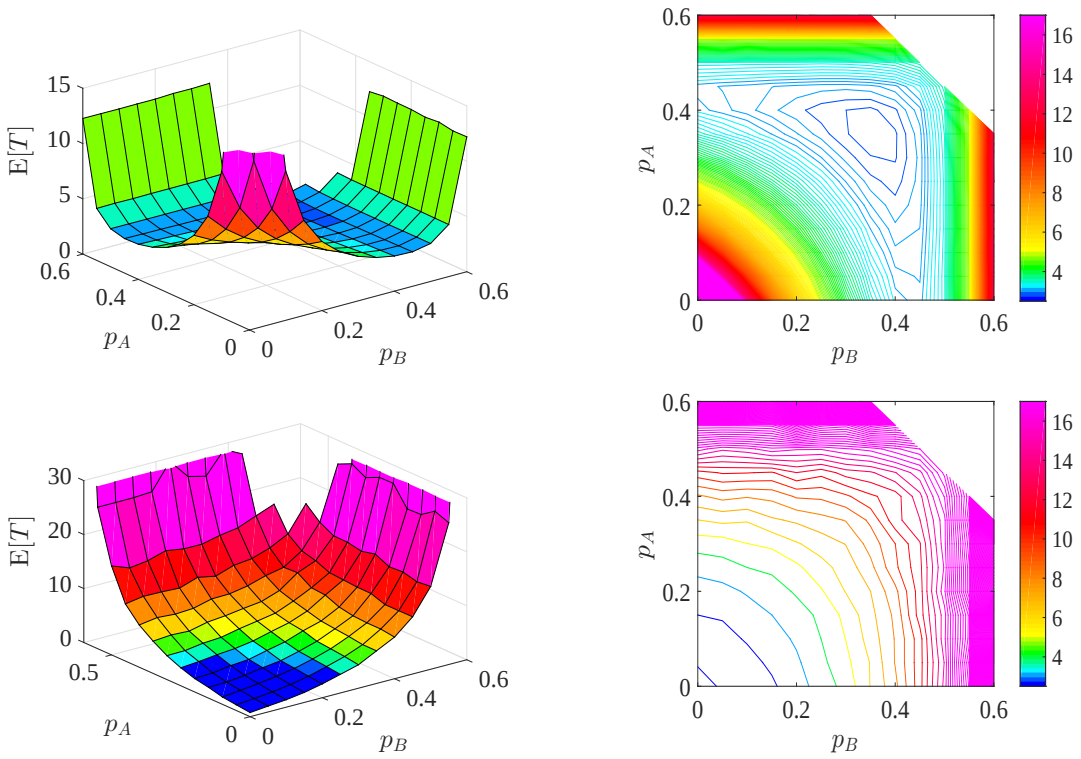


Figure 12: Mean response time under LRF where $\lambda = 1.6$ and service times follow a two-phase Erlang distribution with mean 1 (top row) and a two-phase Hyperexponential distribution with mean 1 and squared coefficient of variation 10 (bottom row).

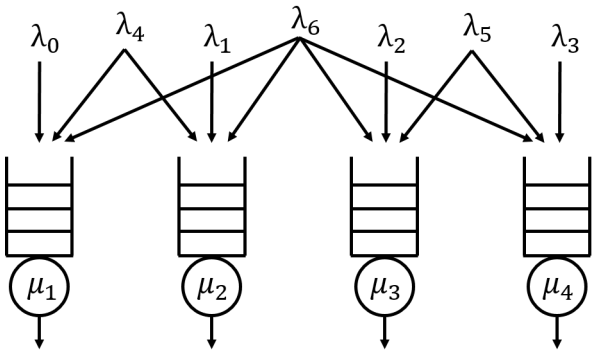
both Erlang and Hyperexponential service times as both p_A and p_B vary; here we can see clearly that none of the monotonicity, convexity, or cross-derivative results hold under Erlang service times.

The observation that whether redundancy helps depends on the service time distribution is consistent with analytical results in the prior work. For example, when job sizes follow a New Worse than Used distribution it is best to make all jobs fully redundant [25]. We conjecture that a similar condition is required for monotonicity and convexity under LRF.

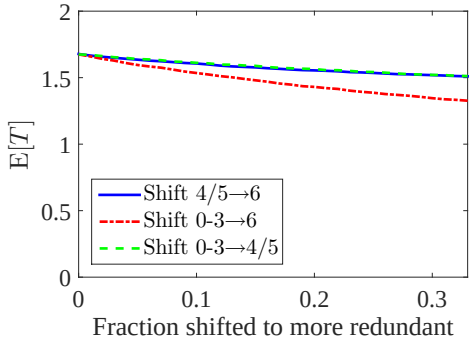
4.4.4. Larger Nested Systems

Thus far we have focused on the $\mathbb{W}$ model, which, as a very small nested system, provides a useful case study for understanding our results. However, all of our results apply to any general nested system. Here we study the system shown in Figure 13, which has four servers and seven job classes with differing degrees of redundancy. We begin by assuming that a third of all jobs are "fixed" at each redundancy degree (i.e., $p_0 + p_1 + p_2 + p_3 = p_4 + p_5 = p_6 = 1/3$) and study the effect of shifting jobs from less-redundant classes to more-redundant classes. This setup allows us to investigate not only the benefit of having more jobs that are redundant, but also the impact of the degree of redundancy. For simplicity, we assume the system is symmetric, meaning that all servers have the same rate and all job classes with the same degree of redundancy have the same arrival rate.

We consider three cases: shifting non-redundant jobs (classes 0, 1, 2, and 3) to being partially redundant (classes 4 and 5), shifting non-redundant jobs to being fully redundant (class 6), and shifting partially redundant jobs to being fully redundant. In all cases, increasing the fraction of jobs that are more redundant leads to a decrease in overall mean response time. The largest impact comes from shifting non-redundant jobs to being fully redundant: the previously non-redundant jobs benefit from their increased redundancy, and class-4 and 5 jobs benefit significantly because they no longer have to wait behind any non-redundant jobs. Interestingly, the impact of shifting partially redundant jobs to being fully redundant is the same as that of shifting non-redundant jobs to being partially redundant. This suggests that increasing the overall



(a) A larger nested system



(b) Effect of making some jobs more redundant

Figure 13: (a) A nested system with four servers and seven job classes. (b) Mean response time under LRF in the system at left, where each server has rate 1 and $\lambda = 3.2$. At all times we hold $p_0 = p_1 = p_2 = p_3$ and $p_4 = p_5$. In the baseline system, $p_0 + p_1 + p_2 + p_3 = p_4 + p_5 = p_6 = 1/3$, and we study the effect of shifting class 4 and 5 job to class 6 (solid blue line), shifting class 0,1,2, and 3 jobs to class 6 (dot-dashed red line), and shifting class 0, 1, 2, and 3 jobs to classes 4 and 5 (dashed green line).

amount of redundancy in the system is more important than precisely where in the system the redundancy is added.

In practice, data centers typically are even larger, consisting of many hundreds or thousands of servers. The "medium-sized" system studied here represents a useful case study for understanding how the system response to varying the redundancy degree among several possible levels. The lessons learned in this smaller system are likely to translate to even larger, more realistically sized redundancy systems.

5. Convexity Under FCFS and PF

We now turn to scheduling policies other than LRF. Apart from the change in scheduling discipline at each server, the model otherwise remains as defined in Section 3. In Section 5.1 we consider First Come First Served (FCFS) scheduling, and in Section 5.2 we consider Primaries First (PF) scheduling.

5.1. FCFS

We have previously derived exact closed-form results for per-class and overall mean response time under FCFS [16]. We define $\mathcal{I}_i$ to be the subsystem in which class-i jobs are fully redundant (this subsystem includes the servers in S_i and the job classes j such that $S_j \subseteq S_i$). Let $\rho_i = \frac{\lambda_i}{\mu_{\mathcal{I}_i} - \lambda_{\mathcal{I}_i} + \lambda_i}$. For completeness, we repeat Theorem 2 of [16] as Theorem 3 here; we also rephrase the theorem slightly because [16] deals with the Laplace transform of per-class response time, whereas here we focus on the mean. Note that this result requires Poisson arrivals.

Theorem 3. *In a nested redundancy system with Poisson arrivals and FCFS scheduling, the response time of class-i jobs is*

$$\mathbb{E}[T_i] = \frac{1}{\mu_{\mathcal{I}_i} - \lambda_{\mathcal{I}_i}} + \sum_{j:S_i \subset S_j} \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}}. \tag{9}$$

The overall system mean response time is

$$\mathbb{E}[T] = \sum_i p_i \mathbb{E}[T_i].$$

Surprisingly, the monotonicity and convexity results that we proved under LRF do *not* necessarily hold under FCFS. The overall mean response time actually can *increase* as the fraction of jobs that are redundant increases. We can understand this behavior by looking more closely at the per-class mean response times as p_R increases. Proposition 1 follows immediately from the per-class mean response times given in Theorem 3.

Proposition 1. *Consider a nested redundancy system with Poisson arrivals and FCFS scheduling, and let classes A, B, and R be such that $S_A \cap S_B = \emptyset$, $S_A \subset S_R$, and $S_B \subset S_R$. Then as p_A decreases and p_R increases, holding $p_A + p_R$ and all other class probabilities constant:*

1. $\mathbb{E}[T_R]$ *is constant.*
2. $\mathbb{E}[T_A]$ *is decreasing and convex.*
3. $\mathbb{E}[T_B]$ *is increasing and concave.*

Proof. 1. From Theorem 3, we have

$$\mathbb{E}[T_R] = \frac{1}{\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R}} + \sum_{j:S_R \subset S_j} \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}}.$$

This is constant in p_R since $\lambda_{\mathcal{I}_R}$ and all $\lambda_{\mathcal{I}_j}$ terms include in the sum the term $\lambda_A + \lambda_R$.

2. Let $c = p_A + p_R$. From Theorem 3, we have

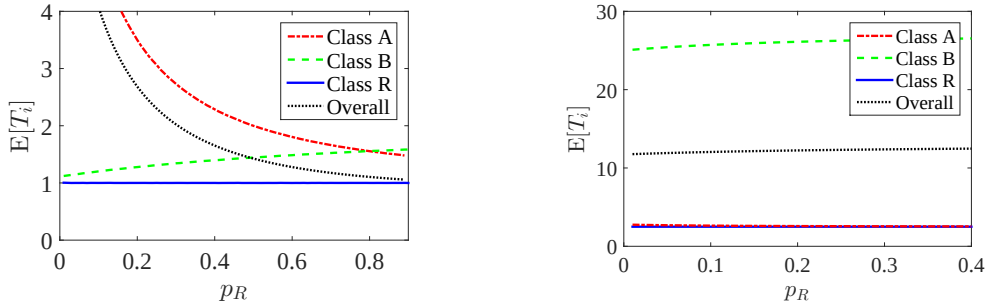
$$\mathbb{E}[T_A] = \frac{1}{\mu_{\mathcal{I}_A} - \lambda_{\mathcal{I}_A \setminus A} - \lambda(c - p_R)} + \sum_{j:S_A \subset S_j \subset S_R} \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}} + \frac{\rho_R}{\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R}} + \sum_{j:S_A \subset S_R \subset S_j} \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}}. \quad (10)$$

All terms in the last summation are constant in p_R, and all terms in the first summation are decreasing and convex (which is easily verified by taking the first and second derivatives). The first term is also decreasing and convex, but the third term is increasing and concave so we will consider the first and third terms together. Let

$$Y = \frac{1}{\mu_{\mathcal{I}_A} - \lambda_{\mathcal{I}_A \setminus A} - \lambda(c - p_R)} + \frac{\rho_R}{\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R}} = \frac{1}{\mu_{\mathcal{I}_A} - \lambda_{\mathcal{I}_A \setminus A} - \lambda(c - p_R)} + \frac{\lambda p_R}{(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R} + \lambda p_R)(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R})}.$$

We have $\frac{dY}{dp_R} = \frac{\lambda}{Y_1^2} - \frac{\lambda}{Y_2^2}$ and $\frac{d^2Y}{dp_R^2} = 2\lambda^2(\frac{1}{Y_2^3} - \frac{1}{Y_1^3})$, where $Y_1 = \mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R} + \lambda p_R$ and $Y_2 = \mu_{\mathcal{I}_A} - \lambda_{\mathcal{I}_A \setminus A} - \lambda(c - p_R)$. If $\lambda_{\mathcal{I}_R} - \lambda_{\mathcal{I}_A} - \lambda p_R < \mu_{\mathcal{I}_R} - \mu_{\mathcal{I}_A}$, $\frac{dY}{dp_R}$ is negative and $\frac{d^2Y}{dp_R^2}$ is positive, and hence Y and $\mathbb{E}[T_A]$ are decreasing and convex.

If instead $\lambda_{\mathcal{I}_R} - \lambda_{\mathcal{I}_A} - \lambda p_R > \mu_{\mathcal{I}_R} - \mu_{\mathcal{I}_A}$ (and so $\frac{dY}{dp_R} > 0$), then there must be some class j such that $S_A \subset S_j \subset S_R$ and $\lambda_j > \mu_{\mathcal{I}_j} - \mu_{\mathcal{I}_{j'}}$, where j' denotes the most redundant class such that $S_A \subseteq S_{j'} \subset S_j$. Let class j be the most redundant such class. Let $Z = \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}}$ be the term corresponding to class j in equation (10). We will show that the first derivative of Z (note that this derivative is negative) has greater magnitude than the derivative of Y, so overall $\mathbb{E}[T_A]$ still has negative derivative, and we will show that



(a) $\lambda = 1, p_B = 0.1$ (b) $\lambda = 1.6, p_B = 0.6$

Figure 14: Per-class mean response times under FCFS as a function of p_R when $\mu_1 = \mu_2 = 1$. Here p_B is held constant and p_A decreases as p_R increases.

the second derivative of Z (which is positive) has greater magnitude than the second derivative of Y, so overall $\mathbb{E}[T_A]$ still has positive second derivative. We have $\frac{dZ}{dp_R} = \frac{\lambda}{Z_1^2} - \frac{\lambda}{Z_2^2}$ and $\frac{d^2Z}{dp_R^2} = 2\lambda^2(\frac{1}{Z_2^3} - \frac{1}{Z_1^3})$, where $Z_1 = \mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j \setminus A} - \lambda(c - p_R) + \lambda p_j$ and $Z_2 = \mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j \setminus A} - \lambda(c - p_R)$. Comparing Y_2 and Z_1, we find that $Z_1 > Y_2$ since $\mu_{\mathcal{I}_j} > \mu_{\mathcal{I}_A}$. Comparing Y_1 and Z_2, we find that $Y_1 > Z_2$ if $\mu_{\mathcal{I}_R} - \mu_{\mathcal{I}_j} > \lambda_{\mathcal{I}_R} - \lambda_{\mathcal{I}_j} - \lambda p_R$. This is equivalent to saying that we need $\sum_{i:S_j \subset S_i \subseteq S_R}(\mu_{\mathcal{I}_i} - \mu_{\mathcal{I}_{i'}}) > \sum_{i:S_j \subset S_i \subset S_R} \lambda_i$, which must be true since j is the most redundant class with $\lambda_j > \mu_{\mathcal{I}_j} - \mu_{\mathcal{I}_{j'}}$.

Putting this together, we have

$$\frac{dY}{dp_R} + \frac{dZ}{dp_R} = \lambda\left(\frac{1}{Z_1^2} - \frac{1}{Y_2^2} + \frac{1}{Y_1^2} - \frac{1}{Z_2^2}\right) < 0,$$
$$\frac{d^2Y}{dp_R^2} + \frac{d^2Z}{dp_R^2} = 2\lambda^2\left(\frac{1}{Z_2^3} - \frac{1}{Y_1^3} + \frac{1}{Y_2^3} - \frac{1}{Z_1^3}\right) > 0,$$

so $\frac{d\mathbb{E}[T_A]}{dp_R} < 0$, $\frac{d^2\mathbb{E}[T_A]}{dp_R^2} > 0$, and $\mathbb{E}[T_A]$ is decreasing and convex.

3. From Theorem 3, we have

$$\mathbb{E}[T_B] = \frac{1}{\mu_{\mathcal{I}_B} - \lambda_{\mathcal{I}_B}} + \sum_{\substack{j:S_B \subset S_j \\ j \neq R}} \frac{\rho_j}{\mu_{\mathcal{I}_j} - \lambda_{\mathcal{I}_j}} + \frac{\rho_R}{\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R}}.$$

All terms except the last are constant in p_R, hence the only relevant term is

$$X = \frac{\rho_R}{\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R}} = \frac{\lambda p_R}{(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R} + \lambda p_R)(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R})}.$$

The first derivative of X is

$$\frac{dX}{dp_R} = \frac{\lambda}{(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R} + \lambda p_R)^2} > 0,$$

so X and hence $\mathbb{E}[T_B]$ is increasing in p_R. The second derivative of X is

$$\frac{d^2X}{dp_R^2} = \frac{-2\lambda^2}{(\mu_{\mathcal{I}_R} - \lambda_{\mathcal{I}_R} + \lambda p_R)^3} < 0,$$

so X and hence $\mathbb{E}[T_B]$ is concave in p_R. □

Since $\mathbb{E}[T_A]$ is increasing in p_R and $\mathbb{E}[T_B]$ is decreasing, overall mean response time could either increase or decrease. Figure 14(b) shows an example of the circumstances under which overall mean response time can increase, breaking down the overall response time by class. As under LRF, here we consider the $\mathbb{W}$ model with service rate $\mu = 1$ at each server. In this example, the overall arrival rate is high ($\lambda = 1.6$) and the fraction of jobs that are class-B is high ($p_B = 0.6$) so the class-B load on server 2 is very high ($\rho_B = \lambda_B/\mu = 0.96$). As p_R increases (holding $p_A + p_R$ constant), the load on server 2 increases even further, so the class-B mean response time, $\mathbb{E}[T_B]$, increases. Moreover, the marginal impact on $\mathbb{E}[T_B]$ decreases as p_R increases further because most of the newly-redundant class-R jobs end up being served on server 1. Hence $\mathbb{E}[T_B]$ is concave in p_R. Consistent with analytical results [16], $\mathbb{E}[T_R]$ does not change as a function of p_R; $\mathbb{E}[T_A]$ is relatively unaffected in this example. Since class-B jobs comprise a large proportion of all jobs, the behavior of $\mathbb{E}[T_B]$ dominates the overall mean response time, causing the overall mean response time to be increasing and concave in p_R.

In certain special cases, the monotonicity and convexity results hold under FCFS. One such special case is a symmetric system, in which all servers are identical with respect to the number of different job classes they serve and the redundancy degrees of those classes. Furthermore, in a symmetric system all classes i that have the same $|S_i|$ also have the same p_i, and as we increase the proportion of more redundant jobs, we decrease the proportion of all less redundant classes equally. For example, in the $\mathbb{W}$ model, holding

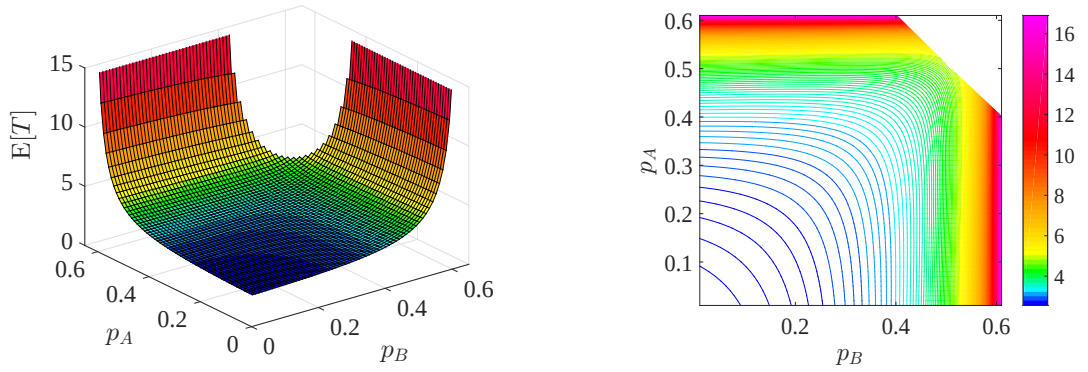


Figure 15: Mean response time under FCFS as a function of p_A and p_B when $\lambda = 1.6$ and $\mu_1 = \mu_2 = 1$.

$\lambda = \lambda_A + \lambda_B + \lambda_R$ and $\mu = \mu_1 + \mu_2$ fixed, letting $\lambda_A = \lambda_B = (\lambda - \lambda_R)/2$ and $\mu_1 = \mu_2 = \mu/2$ yields a symmetric system. In this case, the response times in equation (9) become $\mathbb{E}[T_R] = \frac{1}{\mu-\lambda}$ and $\mathbb{E}[T_A] = \mathbb{E}[T_B] = \frac{1}{\mu/2-(\lambda-\lambda_R)/2} + \frac{\lambda_R}{(\mu-\lambda+\lambda_R)(\mu-\lambda)}$, the first of which is constant and the second of which is decreasing and convex in λ_R. This result can be extended to more general symmetric systems.

Under FCFS, as under LRF, and regardless of whether the system is symmetric, the cross-derivative shows an increasing marginal benefit of redundancy: in any nested system, given two classes A and B such that $S_A \cap S_B = \emptyset$ and a third class R such that $S_A \subset S_R$ and $S_B \subset S_R$, as more class-B jobs shift to becoming class-R jobs, the marginal impact of shifting additional class-A jobs to class-R increases. Let $\mathbb{E}[T(p_A, p_B)]$ denote the mean response time in a system in which p_A fraction of the jobs are class-A and p_B fraction of the jobs are class-B, holding $p_A + p_B + p_R$ constant.

Theorem 4. *Consider a nested redundancy system with Poisson arrivals and FCFS scheduling, and let classes A, B, and R be such that $S_A \cap S_B = \emptyset$, $S_A \subset S_R$, and $S_B \subset S_R$. Then*

$$\mathbb{E}[T(p_A, p_B)] - \mathbb{E}[T(p_A - \epsilon, p_B)] < \mathbb{E}[T(p_A, p_B - \delta)] - \mathbb{E}[T(p_A - \epsilon, p_B - \delta)].$$

That is, as a greater fraction of class-B jobs become redundant, the marginal benefit of shifting jobs from class-A to class-R increases.

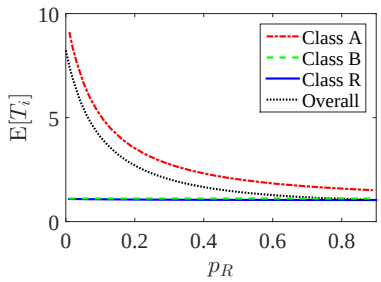
Proof. The proof follows immediately from the exact, closed-form expression for $\mathbb{E}[T]$ given in Theorem 3; we omit the details. □

The contour plot in Figure 15 illustrates this result. As under LRF, when p_B is high the effect of shifting jobs from class-A to class-R is much smaller than when p_B is low.

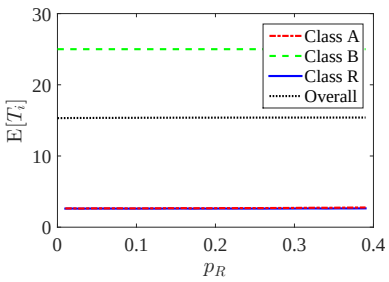
5.2. Primaries First

Under Primaries First (PF) scheduling, each job designates one of its copies as its primary copy, and all other copies (if any) are designated secondary copies. At each server, primaries are given strict preemptive priority over secondaries; within the primaries (respectively, secondaries) all jobs are served in FCFS order regardless of class. In defining PF, we also must specify which copy of a redundant job is designated its primary. In [16], we studied the effect of shifting some jobs from class-A to class-R in the $\mathbb{W}$ model, assuming that all class-R primaries are at server 1. For consistency with the prior work we adopt the same definition of PF here: for our numerical results in the $\mathbb{W}$ model, all primary copies are at server 1.

In [16] we considered the impact of introducing redundancy to a system with no redundancy. Here we want to consider the effect of increasing the proportion of customers that are redundant. We start in Syst(ϵ_A), which consists of $p_A + \epsilon$ class-A jobs, p_B class-B jobs, and p_R class-R jobs, and shift an ϵ fraction of jobs from class-A to class-R so that our new system, Syst(ϵ_R), has p_A class-A jobs, p_B class-B jobs, and $p_R + \epsilon$ class-R jobs (with $p_A + p_B + p_R + \epsilon = 1$). To preserve fairness, we assume that the secondary copies of the ϵ jobs, which shift from being class-A to class-R, have lowest preemptive priority among all jobs at



(a) $\lambda = 1, p_B = 0.1$



(b) $\lambda = 1.6, p_B = 0.6$

Figure 16: Per-class mean response times under PF as a function of p_R when $\mu_1 = \mu_2 = 1$. Here $p_B = 0.1$ is held constant, and p_A decreases as p_R increases.

server 2. This is consistent with our earlier definition of PF scheduling. Let $N_i^{(\epsilon_j)}(t)$ be the number of class i customers at time t in Syst(ϵ_j), $i = A, B, R, \epsilon$, $j = A, R$, so, e.g., $N_A^{(\epsilon_A)}(t)$ and $N_A^{(\epsilon_R)}(t)$ count the number of jobs that are class-A in both systems. Then we have the following, which shows that all customers are better off when redundancy increases under PF scheduling.

Proposition 2. *For an arbitrary exogenous arrival process and exponential service times,*

$$\{N_A^{(\epsilon_R)}(t), N_B^{(\epsilon_R)}(t), N_R^{(\epsilon_R)}(t), N_\epsilon^{(\epsilon_R)}(t)\}_{t=0}^{\infty} \leq \{N_A^{(\epsilon_A)}(t), N_B^{(\epsilon_A)}(t), N_R^{(\epsilon_A)}(t), N_\epsilon^{(\epsilon_A)}(t)\}_{t=0}^{\infty}.$$

Proof. We consider the argument for a single ϵ job; it is easily extended to a fixed proportion, ϵ, as in Corollaries 1 and 2 for LRF. Note that the ϵ job has no effect on the class-B jobs, which experience server 2 as if they were the only jobs in both systems, i.e., $N_B^{(\epsilon_R)}(t) = N_B^{(\epsilon_A)}(t)$, jointly for all t. If the primary copy of the ϵ job (which is at server 1) finishes before its secondary copy in Syst(ϵ_R), then the ϵ job finishes at the same time in both systems, and $N_i^{(\epsilon_R)}(t) = N_i^{(\epsilon_A)}(t)$ jointly for all t and $i = A, B, R, \epsilon$. Otherwise, if the secondary copy of the ϵ job finishes first in Syst(ϵ_R) (let time τ be its completion time), then $N_\epsilon^{(\epsilon_R)}(t) = N_\epsilon^{(\epsilon_A)}(t)$ jointly for all $t < \tau$, and $N_\epsilon^{(\epsilon_R)}(t) < N_\epsilon^{(\epsilon_A)}(t)$ jointly for all $t \geq \tau$. After time τ, class-A and class-R jobs are "worse off" in Syst(ϵ_A) because of the extra ϵ job at server 1, which will delay service completions for other jobs. Hence $N_A^{(\epsilon_R)}(t) \leq N_A^{(\epsilon_A)}(t)$ and $N_R^{(\epsilon_R)}(t) \leq N_R^{(\epsilon_A)}(t)$ jointly for all time t. □

Proposition 2 tells us that under PF, all classes of jobs, included the shifted jobs, are better off as jobs shift to become more redundant.

In Figure 16 we see that, as Proposition 2 tells us, class-B jobs are indifferent to increasing redundancy—regardless of λ—because all of the secondary copies at server 2 have lower preemptive priority than the class-B jobs. Class-A jobs benefit from increasing redundancy, but this benefit is smaller than under LRF because under LRF class-A jobs get full preemptive priority over class-R jobs, whereas under PF the primary copies of redundant jobs wait in FCFS order at server 1. We note that the class-R curve in Figure 16 includes both class-R and shifted ϵ jobs; this combined set of jobs sees a small decrease in mean response time as p_R increases because a small number of class-R jobs have their secondary copies complete service at server 2 before their primary copies complete at server 1; as p_R increases more jobs have the opportunity to benefit from waiting at both servers. Hence, PF is fair in the sense that no class is harmed by increasing redundancy.

Figure 17 illustrates overall mean response time under PF in the $\mathbb{W}$ model with service rate $\mu = 1$ at both servers. Unlike LRF and FCFS, mean response time is *not* symmetric in p_A and p_B under PF because of how we chose which copy of a class-R job to designate as primary. This asymmetry is visible in the left-hand side of Figure 17. When p_B is high, mean response time changes very little as p_R increases (so p_A decreases). On the other hand, when p_A is high and p_R increases (so p_B decreases), mean response time is concave and at first increases but then decreases. This is because when the class-A load is high, shifting jobs from class-B to class-R (and making the copies on server 1 primary) slightly increases the load at server 1, thereby hurting the class-A jobs, and the class-R jobs become low-priority jobs on server 2 (where their secondaries are located), thereby hurting the class-R jobs. But once all class-B jobs have become class-R

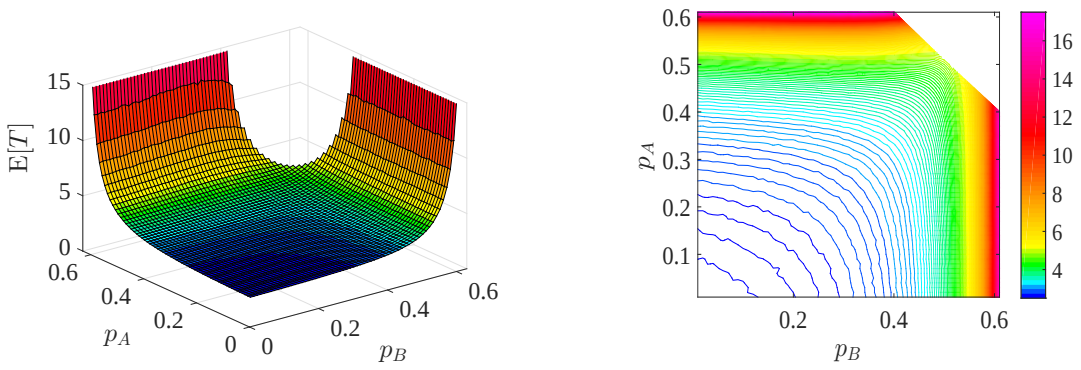


Figure 17: Mean response time under PF as a function of p_A and p_B when $\lambda = 1.6$ and $\mu_1 = \mu_2 = 1$.

jobs, the class-R experience is effectively the same as when all class-R jobs were class-B; mean response time decreases again to reach this point.

While it is difficult to prove convexity under PF, our numerical results suggest that mean response time is convex in p_R, assuming that p_B is held constant and that the copies for redundant jobs are primary at server 1.

Conjecture 2. *In a system with Poisson arrivals and exponential service times and PF scheduling, when p_B is held constant, overall mean response time is convex in p_R.*

Figure 17 supports this conjecture. We further conjecture that a similar cross-derivative result as under LRF and FCFS also holds under PF.

Conjecture 3. *In a system with Poisson arrivals and exponential service times and PF scheduling, as an increasing fraction of jobs shifts from class-B to class-R (where these jobs' primary copies are at server 2), the marginal effect of shifting jobs from class-A to class-R (where these jobs' primary copies are at server 1) increases.*

Figure 17 also supports Conjecture 3. The contour plot in Figure 17 shows that when p_B is high, shifting jobs from class-A to class-R has relatively little effect, whereas when p_B is low substantial gains are possible from making some class-A jobs redundant.

6. Conclusion

This paper studied the marginal effects of increasing redundancy: how much redundancy truly is needed in order to achieve a significant response time improvement? We investigated this question under three different scheduling policies, Least Redundant First, First Come First Served, and Primaries First, and found that the answer depends on the scheduling policy. One of our primary contributions is a proof that LRF, which we previously showed is optimal with respect to minimizing mean response on any sample path, yields mean response times that are monotonically decreasing and convex in the fraction of jobs that are redundant. That is, under LRF scheduling more redundancy is better, but the biggest gains come from adding only a small amount of redundancy to the system. Our numerical results indicate that this is also true under PF scheduling. However, redundancy is not always guaranteed to improve response time, nor is more redundancy necessarily better. In contrast to LRF and PF, under FCFS scheduling increasing the fraction of jobs that are redundant can actually *increase* mean response time, and when redundancy does help the improvement is not necessarily convex. This surprising behavior occurs when one server experiences a high load due to non-redundant jobs, and jobs of a different class shift to being redundant on that server. We also studied cross-derivative effects and found that under all three policies, the marginal impact of making jobs of a particular class more redundant *increases* in the fraction of jobs of a different class that have become more redundant. One important implication of this result is that *symmetric systems yield lower response times.*

Our results show that even in a system with i.i.d. exponential service times, redundancy is not a guaranteed win. Instead, scheduling plays an important role in whether redundancy helps or hurts. Scheduling policies that are likely to be successful share the common feature that they *defer redundancy until the system is otherwise idle*. Under LRF, this is accomplished by giving less-redundant jobs preemptive priority over more-redundant jobs. Under PF, this is accomplished by giving extra copies (regardless of redundancy level) lowest priority. In both cases, the effect is that a server will only work on a redundant copy of a job when that server's queue is empty of non-redundant jobs (or primary copies). By "protecting" non-redundant jobs from having to wait behind redundant copies, LRF and PF allow redundancy to always be a win. In contrast, FCFS interleaves non-redundant jobs with redundant copies, meaning that a non-redundant job may have to wait behind (potentially many) copies that could be served elsewhere.

The work in this paper focuses on the i.i.d. exponential case, but the lessons learned from our results have broader implications for how to schedule jobs in general redundancy systems, where service times may not be exponentially distributed and may not be independent across an individual job's copies. In the i.i.d. exponential setting, redundancy does not add work to the system, so it is particularly striking that it can nonetheless hurt overall response time. The costs of redundancy will be even more pronounced in systems in which redundancy *can* add work to the system. In such systems, it is even more important to ensure that non-redundant jobs are protected from the potentially harmful effects of waiting behind other jobs' redundant copies. The observation that it is best to defer redundancy until servers are otherwise idle likely will be even more crucial in aiding the design of effective scheduling policies for this setting. The results we present in this paper represent a strong foundation on which to build an even deeper understanding of the interaction between scheduling and redundancy.

Acknowledgments

We would like to thank the anonymous reviewers for their detailed feedback, which helped us to greatly improve the paper.

References

[1] Adan, I. and G. Weiss. (2014). A skill based parallel service system under FCFS-ALIS—steady state, overloads, and abandonments. Stochastic Systems 4(1): 250-299.

[2] Adan, I., I. Klener, R. Righter, and G. Weiss. (2018). FCFS parallel service systems and matching models. Performance Evaluation, to appear.

[3] Ahn, H.-S. and R. Righter. (2005). Multi-actor Markov decision processes. Journal of Applied Probability 42: 15.26.

[4] Akgun, O., R. Righter, and R. Wolff. (2012). Understanding the Marginal Impact of Customer Flexibility, Queueing Systems, vol. 71, pp. 5-23.

[5] Akgun, O., R. Righter, and R. Wolff. (2012). Partial flexibility in routing and scheduling. Advances in Applied Probability 45: 637-691.

[6] Aksin, O. Z., Karaesmen, F., Ormeci, E. L. (2007). A review of workforce cross-training in call centers from an operations management perspective. In Workforce Cross Training Handbook, ed. D. Nembhard. CRC Press.

[7] Ananthanarayanan, G., A. Ghodsi, S. Shenker, and I. Stoica. (2013). Effective straggler mitigation: Attack of the clones. Proceedings of the 10'th USENIX Symposium on Networked Systems Design and Implementation. 185-198.

[8] Ananthanarayanan, G., M.C.-C. Hung, X. Ren, I. Stoica, A. Wierman, and M. Yu. (2014). GRASS: Trimming stragglers in approximation analytics. Proceedings of the 11'th USENIX Symposium on Networked Systems Design and Implementation. 289-302.

[9] Ayesta, U., T. Bodas, and I.M. Verloop. (2018). On redundancy-d with cancel-on-start a.k.a. Join-shortest-work(d). MAMA Workshop, SIGMETRICS.

[10] Ayesta, U., T. Bodas, and I.M. Verloop. (2018). On a unifying product form framework for redundancy models. IFIP Performance.

[11] Bassamboo, A., R.S. Randhawa, J.A. van Mieghem. (2012). A little flexibility is all you need: On the asymptotic value of flexible capacity in parallel queueing systems. Operations Research. 60: 1423-1435.

[12] Bonald, T., and C. Comte. (2017). Balanced fair resource sharing in computer clusters. Performance Evaluation, 116, 70-83.

[13] Bonald, T., C. Comte, and F. Mathieu. (2017). Performance of balanced fairness in resource pools: A recursive approach. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 1(2), 41.

[14] Chen, X., J. Zhang, and Y. Zhou. (2015). Optimal sparse designs for process flexibility via probabilistic expanders. Operations Research. 63: 1159-1176.

[15] Dobber, M, R. van der Mei, and G. Koole. (2009). Dynamic load balancing and job replication in a global-scale grid environment: A comparison. IEEE Transactions on Parallel and Distributed Systems. 20: 207-218.

[16] Gardner, K., M. Harchol-Balter, E. Hyytiä, and R. Righter. (2017). Scheduling for efficiency and fairness in systems with redundancy. Performance Evaluation, 116: 1-25.

[17] Gardner, K., M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, and S. Zbarsky. (2017). Redundancy-d: The power of d choices for redundancy. Operations Research 65(4), pp. 1078-1094.

[18] Gardner, K., S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytiä, and Alan Scheller-Wolf. Reducing latency via redundant requests: Exact analysis. In SIGMETRICS, June 2015.

[19] Joshi, G., Y. Liu, and E. Soljanin. (2012). Coding for fast content download. In Allerton Conference'12, pp. 326-333.

[20] Graves, S. C., Tomlin, B. T. (2003). Process flexibility in supply chains. Manag. Sci. 49: 907-919.

[21] He, Y.T., Down, D. (2009). On accommodating customer flexibility in service systems. INFOR. 47: 289-295.

[22] Hopp, W. J., Tekin, E., van Oyen, M. P. (2004). Benefits of skill chaining in serial production lines with cross-trained workers. Manag. Sci. 50: 83-98.

[23] Hopp, W. J., van Oyen, M. P. (2004). Agile workforce evaluation: A framework for crosstraining and coordination. IIE Trans. 36: 919–940.

[24] Jordan, W. C., Graves, S.C. (1995). Principles on the benefits of manufacturing process flexibility. Manag. Sci. 41: 577–594.

[25] Koole, G., and R. Righter. (2009) Resource allocation in grid computing. Journal of Scheduling, 11: 163-173.

[26] Nageswaran, L., and A.A. Scheller-Wolf. (2017). Queues with Redundancy: Is Waiting in Multiple Lines Fair? Available at SSRN: http://dx.doi.org/10.2139/ssrn.2833549.

[27] Jung, D., S.H. Chin, K.S. Chung, T. Suh, H.C. Yu, J.M. Gil. (2010). An effective job replication technique based on reliability and performance in mobile grids. In: International Conference on Grid and Pervasive Computing, pp. 47-58.

[28] Mitzenmacher, M. (2001). The power of two choices in randomized load balancing. IEEE Trans. Parallel. Distrib. Syst. 12: 1094-1104.

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

[30] Sun, Y., C.E. Koksal, and N.B. Shroff. (2016). On delay-optimal scheduling in queueing systems with replications. arXiv preprint arXiv:1603.07322.

[31] Sun, Y., Z. Zheng, C.E. Koksal, K.-H. Kim, and N.B Shroff. (2015). Provably delay efficient data retrieving in storage clouds. In 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 585-593.

[32] Tsitsiklis, J.N., Xu, K. (2011). On the power of (even a little) centralization in distributed processing. Proc. ACM SIGMETRICS, San Jose.

[33] Tsitsiklis, J. and K. Xu. (2017). Flexible Queueing Architectures. Operations Research 65(5): 1398-1413.

[34] Turner, S.R.E. (1998). The effect of increasing routing choice on resource pooling. Probab. Eng. Inf. Sci 12: 109-124.

[35] Van Oyen, M.P., E.G.S. Gel, and W.J. Hopp. (2001). Performance opportunity for workforce agility in collaborative and noncollaborative work systems. IIE Trans. 33: 761-777.