Analyzing Response Time in the Redundancy-d System

Kristen Gardner[1] Samuel Zbarsky[2]
Mark Velednitsky[3] Mor Harchol-Balter[1]
Alan Scheller-Wolf[4]

April 2016
CMU-CS-15-141R

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Mathematics Department, Carnegie Mellon University, Pittsburgh, PA, USA
[3]IEOR Department, UC Berkeley, Berkeley, CA, USA
[4]Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA

This work was supported by NSF-CMMI- 1538204, NSF-CMMI-1334194, and NSF-CSR-1116282, by the Intel Science and Technology Center for Cloud Computing, by a Google Faculty Research Award 2015/16, and by the National Physical Science Consortium.

Keywords: Redundancy, Markov chains, differential equations, server farms, dispatching

Abstract

Redundancy is an important strategy for reducing response time in multi-server queueing systems that has been used in a variety of settings, but only recently has begun to be studied analytically. The idea behind redundancy is that customers can greatly reduce their response time by waiting in multiple queues at the same time, thereby experiencing the minimum time across queues. Redundancy has been shown to produce significant response time improvements in applications ranging from organ transplant waitlists to Google's BigTable service. However, despite the growing body of theoretical and empirical work on the benefits of redundancy, there is little work addressing the questions of how many copies one needs to make in order to achieve a response time benefit, and the magnitude of the potential gains.

In this paper we propose a model to evaluate these questions. Our model is called the Redundancy-$\mathbf{d}$ system; each incoming job makes copies at a constant number of servers, $\mathbf{d}$, chosen at random. We derive the first exact expressions for mean response time in Redundancy-$\mathbf{d}$ systems with any finite number of servers, as well as expressions for the distribution of response time which are exact as the number of servers approaches infinity, provided an asymptotic independence assumption holds. Using our analysis, we show that the biggest response time improvement comes from having each job wait in only $\mathbf{d} = 2$ queues.

1 Introduction

In 2009, a unique aspect of Steve Jobs's liver transplant made headlines: even though Jobs lived in California, his transplant was performed in Tennessee. Typically, a patient waiting for a deceased donor organ in the U.S. puts his name on the waitlist for the geographic region in which he lives. Jobs did what is known as *multiple listing*: his name appeared on the waitlist in both California and Tennessee, thereby reducing the time he had to wait to receive a transplant when a liver became available sooner in Tennessee than in California. Multiple listing is becoming an increasingly common strategy to reduce the waiting time for deceased donor organ transplants: it allows patients to experience the minimum queueing time across several waitlists. Because of the significant delay reduction that multiple listing offers, services such as OrganJet have begun to facilitate multiple listing at a broad scale ([4]).

The benefits of multiple listing – also called redundancy – are not unique to organ transplant waitlists. In computer systems, redundancy is defined as creating multiple copies of the same job and dispatching these to different servers, waiting for only the first copy to complete. In the context of computer systems, redundancy is useful because server speeds are unpredictable since they depend on external conditions such as garbage collection, network interrupts, or background work. In fact, it has been shown that the *same job* can take much (12 to 27 times) longer to run on one server than another ([1, 18]). For example, in applications such as web page downloads and Google search queries, empirical computer systems work has demonstrated the benefit of using redundancy to minimize both the mean and the tail of response time (e.g., [1, 2, 6, 16]).

One can imagine using redundancy to reduce response time in any system in which there are multiple service providers that are capable of processing each request, but server-dependent variability is significant and unpredictable. There are a myriad of potential applications. When concert tickets first go on sale online people often enlist several friends to try to buy tickets simultaneously in the hopes that one transaction will complete quickly. When placing holds on library books, one can imagine requesting multiple different books at the same time to reduce the time until some book becomes available. Parents trying to enroll their children in daycare may put their names down at multiple daycare centers hoping that some waitlists will end up shorter than others. In all of these cases, redundancy can help customers to achieve shorter waiting times.

However, it often is difficult to know how much redundancy is needed in order to achieve an appreciable benefit. How much faster does a search query complete if it is run on two servers rather than one? Does a patient receive a kidney transplant sooner if she multiple lists in three regions rather than two? What about five? Ten?

In this paper, we study these questions by introducing and analyzing a theoretical model called the Redundancy-$\mathbf{d}$ system (see Figure 1). In the Redundancy-$\mathbf{d}$ system, there are k servers; each arriving job joins the queue at $\mathbf{d}$ of these servers, chosen uniformly at random. Here $\mathbf{d}$ is a constant that does not depend on k, and typically is small relative to k. Each server provides exponential service times with rate μ and works on the jobs in its queue in first-come first-served order. A job may be in service at multiple servers at the same time; in this case its service times are i.i.d. across servers. A job is considered complete as soon as the first copy finishes service, at which time all remaining copies disappear from the system regardless of whether they are in the queue or in service.

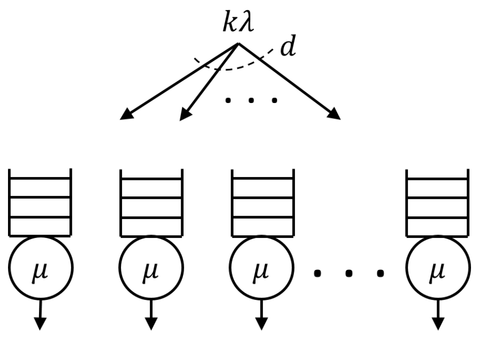


Figure 1: The Redundancy-$\mathbf{d}$ system. The system consists of k servers, each providing exponential service times with rate μ. Jobs arrive to the system as a Poisson process with rate $k\lambda$ and each job sends copies to $\mathbf{d}$ servers chosen uniformly at random. A job is considered complete as soon as the first of its copies completes service.

Our *primary contribution* is providing the first analysis of response time in the Redundancy-$\mathbf{d}$ system. Our analysis follows two approaches. First, we derive an exact closed-form expression for mean response time by modeling the system as a Markov chain with a very detailed state space that tracks the location of all copies of all jobs in the system. The difficulty in finding mean response time for this system lies in aggregating the limiting probabilities for our detailed states, which is necessary to find the distribution of the number of jobs in the system. We present a novel state aggregation approach to accomplish this. We then use generating functions to derive mean response time as a function of the number of servers k and the number of copies per job $\mathbf{d}$ in the Redundancy-$\mathbf{d}$ system (Section 4).

We next turn to the analysis of the *distribution* of response time. For this we need a different approach. We consider the system in the limit as the number of servers k approaches infinity. We make the further assumption that in this asymptotic regime, the work in different queues is independent (such independence has been shown to hold under related policies, for example, Join-the-Shortest-Queue dispatching ([13, 17, 19]). Under these assumptions, we formulate a system of differential equations that describes the evolution of the system. Coming up with the right differential equations is not straightforward because the system has a very complicated departure process: each service completion results in the removal of $\mathbf{d}$ copies from different servers. We use our differential equations to derive an asymptotically exact expression for the distribution of response time (Section 5).

We then use our analytical results to investigate the effect of $\mathbf{d}$ on response time in the Redundancy-$\mathbf{d}$ system (Section 6). This problem is reminiscent of the power-of-$\mathbf{d}$ results which exist in the literature for Join-the-Shortest-Queue dispatching [13] (with no redundancy). While the trends we observe are what one might expect, our exact analysis allows us to quantify the magnitude of these trends for the first time. As $\mathbf{d}$ increases, mean response time decreases, and the biggest improvement comes from adding just a single extra copy of each job ($\mathbf{d} = 2$). For example, at high load, setting $\mathbf{d} = 2$ reduces mean response time by a factor of 6. Our results support the empirical observation that the improvement is even more pronounced in the tail: at high load, setting $\mathbf{d} = 2$ reduces tail response time by a factor of 8. We furthermore show that when $\mathbf{d}$ is high, mean response time drops in proportion to $\frac{1}{\mathbf{d}}$. Leveraging the fact that the largest benefit comes from having a single extra replica ($\mathbf{d} = 2$), we introduce the idea of "fractional $\mathbf{d}$" redundancy, in which each job makes on average between one and two copies. We find that even with fewer than two copies on average, redundancy still provides a significant response time improvement.

While our model assumes exponentially distributed service times, the analytical approach we present in Section 5 applies much more generally. We develop a numerical extension to our analytical approach, which allows us to study the effect of $\mathbf{d}$ on response time under non-exponential service time distributions (Section 6.2).

The remainder of this paper is organized as follows. In Section 2 we review prior work on systems related to the Redundancy-$\mathbf{d}$ system. In Section 3 we introduce our theoretical model and discuss how the model is related to practical applications. Sections 4 and 5 present our analytical results for the mean and distribution of response time respectively. In Section 6 we use our analysis to investigate the impact of the choice of $\mathbf{d}$ on response time. Finally, in Section 7 we conclude.

2 Prior Work

Redundancy in Practice

While Steve Jobs's famous multiple listing was for a liver transplant, multiple listing is most common on kidney transplant waitlists. Consequently most of the empirical work on the impacts of multiple listing for deceased organ donor waitlists focuses on kidney transplants. Multiple listing is an extremely effective strategy for reducing the time it takes for a patient to receive a kidney: patients who multiple list wait on average half as long as those who are listed it only one region ([12]). In addition, multiple listing can help to reduce geographic disparities in waiting times ([3]). Unfortunately, multiple listing is not terribly common. In 2003, only about 6% of kidney patients were multiple listed ([12]). Some patients may be unaware that multiple listing is an option; others may be unable to multiple list because it is too costly to travel to alternative transplant centers. The company OrganJet was founded to combat these problems by providing patients with free flights to remote transplant centers, thereby increasing the proportion of patients who are able to multiple list ([4]).

In computer systems, a great deal of empirical work has demonstrated the benefits of redundancy. One common system in which redundancy is used is MapReduce ([7]). MapReduce jobs consist of several distinct phases of work; each phase involves computing many sub-tasks, all of which must complete before the next phase can begin. Preventing sub-tasks from straggling, or taking unusually long amounts of time to complete, is of utmost importance in reducing the overall computation time across all phases of a job. Many MapReduce implementations incorporate redundancy to reduce the likelihood that sub-tasks become stragglers ([2, 20]). This leads to up to a 30% reduction in mean response time ([2]).

Redundancy also helps reduce the tail of response time, which in many computer systems is an even more important metric than the mean. In database systems, redundancy can reduce the 99.9th percentile of response time by a factor of 8, effectively eliminating all variability in response time ([16]). In Google's BigTable service this number is even higher: sending a small number of redundant requests reduces the 99.9th percentile of response time by a factor of 25 ([6]).

Theoretical Models of Redundancy

While redundancy is becoming an increasingly prevalent strategy for reducing response time in queueing systems, there is very little theoretical work analyzing its performance. In this section we discuss how the Redundancy-$\mathbf{d}$ system, which we propose and analyze, is related to several models existing in the literature.

The (n, k) fork-join system has n servers to which each arriving job sends copies of itself. The job is considered complete when $k \leq n$ of these copies are complete. Unlike in the Redundancy-$\mathbf{d}$ system, in the (n, k) fork-join system each job sends copies to all servers and may need multiple copies to complete. The (n, k) fork-join system was first proposed in [9], and bounds and approximations were derived in [9, 10, 14].

In [15], a variation on the (n, k) fork-join system was proposed in which each job sends copies to $r \leq n$ of the servers and is complete when $k \leq r$ of these copies finish service. Both [14] and [15] study the optimal value of r with respect to minimizing mean response time in both central-queue and distributed-queue variations of this system. The Redundancy-$\mathbf{d}$ system can be seen as a distributed-queue $(n, 1)$ fork-join system with $r = \mathbf{d}$. However neither [14] nor [15] provides any analysis quantifying mean response time as a function of r. Our paper provides the first analysis of response time in such a system.

The scenario in which only one copy of a job needs to complete has been studied in several other papers. For example, [11] studies optimal allocation of jobs to servers in a system where jobs are allowed to run on multiple servers at the same time but only one copy needs to complete. The authors find that for service time distributions with decreasing failure rate it is optimal to send redundant copies of each job to all servers. While [11] makes it clear that more redundancy is better, they never analyze the performance of redundancy as a function of the degree of redundancy.

In [16], approximations for response time are derived for a system where each job sends copies to multiple randomly chosen servers, but unlike in the Redundancy-$\mathbf{d}$ system, extra copies are not cancelled upon completion of the first copy. This no-cancellation assumption greatly simplifies the analysis because as the number of servers grows large, one can view each server as being an independent M/M/1 queue. When extra copies are cancelled, we can no longer view the system as independent M/M/1s.

The closest work to the present work is [8], which considers a general redundancy system where each job has a class that specifies the subset of servers to which it sends copies. The system is modeled as a Markov chain in which the state tracks the classes of all jobs in the system in order of arrival; [8] derives the limiting distribution on this state space. While we show in Section 4 that the Redundancy-$\mathbf{d}$ system can be modeled in this fashion, [8] only finds response time in a few simple two- or three-server systems. More importantly, it is unclear from [8] how to utilize the combinatorially complex limiting distribution to find response time in general systems, including the Redundancy-$\mathbf{d}$ system. We provide this analysis in the present work (see Section 4).

3 Model and Motivation

3.1 Mathematical Model

We consider a k-server system called the Redundancy-$\mathbf{d}$ system, shown in Figure 1. Jobs arrive to the system as a Poisson process with rate $k\lambda$. Upon arrival, each job sends a copy of itself to $\mathbf{d}$ servers chosen uniformly at random without replacement. A job is considered complete as soon as its first copy completes, at which time all remaining copies disappear from the system regardless of whether they are in service or in the queue.

Each server provides exponential service times with rate μ and works on the jobs in its queue in first-come first-served order. A job's service times are i.i.d. across servers; the job may be in service at multiple servers at the same time, in which case it experiences the minimum service time among all servers at which it is in service.

We define the system load to be $\rho = \frac{\lambda}{\mu}$. This is the total arrival rate to the system ($k\lambda$) divided by the maximum service rate of the system ($k\mu$). The system is stable as long as $\rho < 1$ (see Section 4).

Our goal is to analyze response time, T, in the Redundancy-$\mathbf{d}$ system as a function of the arrival rate λ, the service rate μ, the number of servers k, and the degree of redundancy $\mathbf{d}$, to help us understand the role redundancy can play in reducing response time.

3.2 Motivation

In this section we discuss our modeling assumptions in the context of two particular applications of redundancy. While our model does not perfectly match all of the practical intricacies of these applications, it captures many important features of the systems. Our exact analysis of the Redundancy-$\mathbf{d}$ system yields several insights that help us to understand the potential benefits of redundancy in such systems.

In the case of the organ transplant waitlist application, the i.i.d. exponentially distributed service times in our model represent the time for a deceased donor organ to become available, and response time is the time from when a patient joins the waitlist until she receives an organ. It is reasonable to imagine that organs become available in different regions according to independent processes (as deaths occur). A person waiting at the head of the queue in two regions waits for the minimum "service time" across these regions. Our model is a simplification of the organ transplant waitlist system in that we ignore geography-dependent factors by assuming that all regions have the same service rate μ and that patients choose regions uniformly at random, and we do not model abandonment due to patient deaths. Nonetheless, our analysis allows us to assess the incremental response time benefit of multiple listing in additional regions.

In the case of computer systems, the service time represents the time needed to process a job. In many applications, the inherent computation time of a job is very small compared to the time added by external factors at the particular server on which it runs. For example, a web request typically takes roughly 1ms, but can easily take as long as 10ms ([6]). Likewise, a query response time can take up to 27 times longer on a virtual machine with high background load as opposed to on an uncongested virtual machine ([18]). Unfortunately, the server slowdown varies unpredictably and thus cannot be taken into account when making dispatching decisions. We use independent exponentially distributed service times as an approximation of this server-dependent variability. Of course actual service times are not necessarily exponentially distributed, but we assume exponential distributions for analytical tractability. In Section 6.2 we discuss how to extend our results to other service time distributions.

4 Markov Chain Analysis

The purpose of this section is to prove Theorem 1, which gives a simple expression for the mean response time in the Redundancy-$\mathbf{d}$ system with k servers.

Theorem 1. *The mean response time in the Redundancy-*$\mathbf{d}$ *system with* k *servers is*

$$\mathrm{E}[T] = \sum_{i=\mathbf{d}}^{k} \frac{1}{k\mu \frac{\binom{k-1}{\mathbf{d}-1}}{\binom{i-1}{\mathbf{d}-1}} - k\lambda}. \tag{1}$$

The remainder of this section is devoted to proving the above result.

4.1 Alternative System View: Class-Based Redundancy

We define a job's *class* as the set of $\mathbf{d}$ particular servers to which the job sends copies. There are $\binom{k}{\mathbf{d}}$ possible classes; all classes are equally likely since each job chooses its servers uniformly at random. Let λ_{class} denote the arrival rate of any class, where

$$\lambda_{\text{class}} = \frac{k\lambda}{\binom{k}{\mathbf{d}}}.$$

Following [8], our system state is a list of all jobs in the system in the order in which they arrived, where we track the class of each job. We write the state as $(c_m,\ c_{m-1}, \ldots,\ c_1)$, denoting that there are m jobs in the system, c_1 is the class of the oldest job in the system (the first of the m jobs to arrive), and c_i is the class of the ith job in the system in order of arrival. Since the state tracks all jobs in the system in order of arrival, the state information implicitly tracks which jobs are in service at which servers. For example, the oldest job in the system, which has class c_1, must be in service at all $\mathbf{d}$ of its servers.

Once we have defined the notion of a job class and written the system state as defined above, we obtain the following result for the limiting distribution of the state space:

Theorem 2. *In the Redundancy-*$\mathbf{d}$ *system, the limiting probability of being in state* $(c_m,\ c_{m-1}, \ldots,\ c_1)$ *is*

$$\pi_{(c_m, c_{m-1}, \ldots, c_1)} = \mathcal{C} \prod_{j=1}^{m} \frac{\lambda_{\text{class}}}{|S_j|\mu}, \tag{2}$$

where S_j is the set of all servers working on jobs $1, \ldots, j$ ($|S_j|$ is the number of servers in this set) and

$$\mathcal{C} = \prod_{i=d}^{k} \left(1 - \frac{\binom{i-1}{d-1}\lambda}{\binom{k-1}{d-1}\mu}\right)$$

is a normalizing constant.

Proof. The general form of the limiting probabilities given in (2) is an immediate consequence of Theorem 1 in [8]. However the normalizing constant $\mathcal{C}$ is not derived there, and this is the heart of our proof.

Let π_m be the limiting probability that there are m jobs in the system (note that π_m results from aggregating states $(c_m, \cdots, c_1)$ over all possible classes $c_1, \ldots, c_m$). If we number our servers as 1 through k, then the normalizing equation

$$\sum_{m=0}^{\infty} \pi_m = 1,$$

combined with the form given in (2), tells us that

$$\begin{aligned} \mathcal{C} &= \Pr\{\text{all servers are idle}\} \\ &= \Pr\{\text{server } k \text{ idle}\} \cdot \Pr\{\text{server } k-1 \text{ idle} \mid \text{server } k \text{ idle}\} \cdots \Pr\{\text{server 1 idle} \mid \text{servers } 2, \ldots, k \text{ idle}\} \\ &= \Pr\{\text{server } k \text{ idle}\} \cdot \Pr\{\text{server } k-1 \text{ idle} \mid \text{server } k \text{ idle}\} \cdots \Pr\{\text{server } \mathbf{d} \text{ idle} \mid \text{servers } \mathbf{d}+1, \ldots, k \text{ idle}\}, \end{aligned} \tag{3}$$

where the last line is due to the fact that if fewer than **d** servers are busy then no jobs can be present.

To find $\Pr\{\text{server } k \text{ idle}\}$, observe that the time-average number of busy servers in a k-server system is $\rho_k \cdot k$, where $\rho_k = \frac{\lambda}{\mu}$ is the total arrival rate to the system, $k\lambda$, divided by the total service capacity, $k\mu$. Since the system is symmetric in permuting the servers, each server, including server k, has probability $1 - \rho_k$ of being idle.

We still need to find $\Pr\{\text{server } k-\ell \text{ idle} \mid \text{servers } k-\ell+1, \ldots, k \text{ idle}\}$. To do this, we will rewrite the limiting probability given in (2) conditioning on servers $k-\ell+1, \ldots, k$ being idle:

$$\Pr\{\text{system in state } c_m, \ldots, c_1 \mid \text{servers } k-\ell+1, \ldots, k \text{ idle}\} = \\ \begin{cases} 0 & \text{if } n \in S_j \text{ for some } k-\ell+1 \leq n \leq k \\ \dfrac{\mathcal{C}}{P_\ell} \cdot \displaystyle\prod_{j=1}^{m} \frac{\lambda_{\text{class}}}{|S_j|\mu} & \text{otherwise,} \end{cases}$$

where $P_\ell = \Pr\{\text{servers } k-\ell+1, \ldots, k \text{ are idle}\}$. This is exactly the limiting probability of being in state $c_m, \ldots, c_1$ in a system that consists of $k-\ell$ servers and includes only those jobs that do not go to any server n, $k-\ell+1 \leq n \leq k$. The total arrival rate to such a system is $\frac{\binom{k-\ell}{\mathbf{d}}}{\binom{k}{\mathbf{d}}} \cdot k\lambda$. The total service capacity is $(k-\ell)\mu$. Hence the time-average fraction of time a particular server is busy in this system is

$$\rho_{k-\ell} = \frac{\frac{\binom{k-\ell}{\mathbf{d}}}{\binom{k}{\mathbf{d}}} \cdot k\lambda}{(k-\ell)\mu} = \frac{\binom{k-\ell}{\mathbf{d}}}{\binom{k}{\mathbf{d}}} \cdot \frac{\lambda}{\mu} \cdot \frac{k}{k-\ell}.$$

The probability that any server, and in particular server $k-\ell$, is idle in this system – and hence in the original k-server system given that servers $k-\ell+1,\ldots,k$ are idle – is $1-\rho_{k-\ell}$.

Going back to (3), we have

$$\begin{aligned}
\mathcal{C} &= \Pr\{\text{server } k \text{ idle}\}\cdot\Pr\{\text{server } k-1 \text{ idle} \mid \text{server } k \text{ idle}\}\cdots\Pr\{\text{server } \mathbf{d} \text{ idle} \mid \text{servers } \mathbf{d}+1,\ldots,k \text{ idle}\}\\
&= \prod_{\ell=0}^{k-\mathbf{d}}(1-\rho_{k-\ell})\\
&= \prod_{\ell=0}^{k-\mathbf{d}}\left(1-\frac{\binom{k-\ell}{\mathbf{d}}}{\binom{k}{\mathbf{d}}}\cdot\frac{\lambda}{\mu}\cdot\frac{k}{k-\ell}\right)\\
&= \prod_{\ell=0}^{k-\mathbf{d}}\left(1-\frac{\binom{k-\ell-1}{\mathbf{d}-1}}{\binom{k-1}{\mathbf{d}-1}}\cdot\frac{\lambda}{\mu}\right)\\
&= \prod_{i=\mathbf{d}}^{k}\left(1-\frac{\binom{i-1}{\mathbf{d}-1}\lambda}{\binom{k-1}{\mathbf{d}-1}\mu}\right).
\end{aligned}$$

□

The form of the limiting probabilities given in (2) is quite unusual. Although it looks like a product form, it cannot be written as a product of per-class terms or as a product of per-server terms. Example 1 illustrates this.

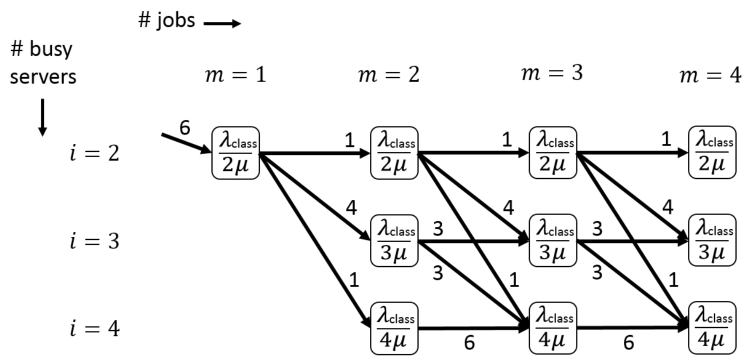
Example 1. *Consider a system with $k=4$ servers and $\mathbf{d}=2$ copies per job. Suppose that there are currently four jobs in the system. The first job has class A and its copies are at servers 1 and 2. The second job has class B and its copies are at servers 2 and 4. The third job has class C and its copies are at servers 3 and 4. The fourth job has class A (the same as the first job) and its copies are at servers 1 and 2. Then the state of the system is (A,C,B,A) and the limiting probability of being in this state is*

$$\pi_{(A,C,B,A)}=\left(\frac{\lambda_{\text{class}}}{4\mu}\right)\left(\frac{\lambda_{\text{class}}}{4\mu}\right)\left(\frac{\lambda_{\text{class}}}{3\mu}\right)\left(\frac{\lambda_{\text{class}}}{2\mu}\right),$$

where the rightmost term is the contribution of the first A arrival and the leftmost term is the contribution of the last A arrival, and where $\lambda_{\text{class}}=\frac{2}{3}\lambda$. Note that the limiting probability is not simply a product of per-class terms or of per-server terms since the denominators depend on the order of all jobs in the system.

Theorem 3. *The Redundancy-$\mathbf{d}$ system is stable when $\rho=\frac{\lambda}{\mu}<1$.*

Proof. The proof of Theorem 3 relies on the state aggregation approach we present in Section 4.2, so we defer the proof to the end of the section. □



# jobs
# busy servers
$m = 1$
$m = 2$
$m = 3$
$m = 4$
$i = 2$
$i = 3$
$i = 4$
$\frac{\lambda_{\text{class}}}{2\mu}$
$\frac{\lambda_{\text{class}}}{3\mu}$
$\frac{\lambda_{\text{class}}}{4\mu}$
6
1
4
3
6

Figure 2: Aggregating states in the $k = 4$, $\mathbf{d} = 2$ system. Horizontally we track the number of jobs in the system and vertically we track the number of busy servers. The value at node (i, m) gives the contribution of the job at position m to the limiting probability. An edge from node (i, m) to node $(j, m + 1)$ has weight equal to the number of classes the job in position $m + 1$ could be in order for there to be j servers busy working on the first $m + 1$ jobs when there were i servers busy working on the first m jobs.

4.2 State Aggregation

One might think that $\mathrm{E}[T]$ follows immediately from the limiting distribution on the state space given in Theorem 2. Unfortunately, knowing the limiting distribution on the state space does not immediately yield results for mean number in system and mean response time. This is because to find mean response time, we must first find

$$\pi_m = \Pr\{m \text{ jobs in system}\}.$$

To do this, we need to sum π values over all $\binom{k}{\mathbf{d}}$ possible classes for each queue position j, $1 \leq j \leq m$. This is not straightforward because the denominators in the limiting probabilities depend on the order of all jobs in the system: $\pi_{(c_m,\ldots,c_1)}$ depends on the particular choices of $c_1, \ldots, c_m$.

The key observation that helps us aggregate states is that we only need to track the denominator contributed to the limiting probability by the job in each queue position j - *not* the specific class c_j of the job. This is equivalent to tracking the number of servers that are busy working on the first j jobs in the queue. We leverage this observation by collapsing our state space so that instead of $\binom{k}{\mathbf{d}}$ possible classes for each position in the queue, we now have at most $k - \mathbf{d}$ possible denominators. In addition, not all denominators are possible for each position; for example, position 1 must contribute denominator $\mathbf{d}\mu$, and if position j contributes denominator $i\mu$ then position $j + 1$ must contribute denominator $\geq i\mu$.

We define $P(i, m)$ to be the limiting probability that there are m jobs in the system and i busy servers, disregarding a normalization constant. To find π_m, we need to compute $P(i, m)$ for all $\mathbf{d} \leq i \leq k$.

At a high level, our approach takes the following steps:

1. Write recurrences for $P(i, m)$, the (unnormalized) limiting probability that there are i servers busy and m jobs in the system (Section 4.2.1).

2. Define a generating function for our recurrences and use this generating function to find $\mathrm{E}[N]$ and $\mathrm{E}[T]$ (Section 4.2.2).

Throughout the remainder of this section we refer to Figure 2, which provides a running example of our approach in the case where $k = 4$ and $\mathbf{d} = 2$.

4.2.1 Formulating Recurrences $P(i, m)$

Our goal in this section is to write recurrences for $P(i, m)$, the (unnormalized) limiting probability that the system has i busy servers and m jobs in the system.

Theorem 4. *For $m > 1$, the limiting probability that there are m jobs in the system and i busy servers satisfies*

$$P(i,m) = \frac{\lambda_{\text{class}}}{\mu i} \cdot \sum_{y=0}^{\mathbf{d}} \binom{i-y}{\mathbf{d}-y} \cdot \binom{k-(i-y)}{y} \cdot P(i-y, m-1). \tag{4}$$

For $m = 1$, we have the initial conditions

$$P(i,1) = \begin{cases} \binom{k}{\mathbf{d}} \frac{\lambda_{\text{class}}}{\mu \mathbf{d}}, & i = \mathbf{d} \\ 0, & \mathbf{d} < i \le k. \end{cases} \tag{5}$$

For $m = 0$, we have the initial conditions

$$P(i,0) = \begin{cases} 1, & i = 0 \\ 0, & i > 0 \end{cases} \tag{6}$$

Proof. We first consider the case $m = 1$. Here there is a single job in the system, so the system state is (c_1). Regardless of the specific class c_1, there are always $\mathbf{d}$ servers busy working on this job and the arrival rate of class c_1 is always λ_{class}, so from Theorem 2 the limiting probability of this state is $\pi_{(c_1)} = \mathcal{C} \cdot \frac{\lambda_{\text{class}}}{\mu \mathbf{d}}$. The job could belong to any class, so there are $\binom{k}{\mathbf{d}}$ states in which $m = 1$. Hence the total probability that there is one job in the system is

$$\pi_1 = \binom{k}{\mathbf{d}} \cdot \mathcal{C} \cdot \frac{\lambda_{\text{class}}}{\mu \mathbf{d}} = \mathcal{C} \cdot P(\mathbf{d}, 1).$$

For any value of $i \neq \mathbf{d}$ it is not possible to have i servers working on only $m = 1$ job, so $P(i, 1) = 0$. This gives the initial conditions in (5) (recall that we omit the normalizing constant).

When $m = 2$, the system state is $(c_2,\ c_1)$. The number of busy servers can range from $\mathbf{d}$ (if both jobs are of the same class and therefore share all $\mathbf{d}$ servers) to $2\mathbf{d}$ (if the two jobs do not share any servers). Hence we need to find expressions for $P(\mathbf{d}, 2),\ P(\mathbf{d}+1, 2), \ldots,\ P(2\mathbf{d}, 2)$.

To find $P(\mathbf{d},2)$, observe that the first job in the system, which has class c_1, contributes a factor of $\frac{\lambda_{\text{class}}}{\mu\mathbf{d}}$ to the limiting probability, and there are $\binom{k}{\mathbf{d}}$ ways of choosing class c_1, so its total contribution to $P(\mathbf{d},2)$ is $\binom{k}{\mathbf{d}}\frac{\lambda_{\text{class}}}{\mu\mathbf{d}}$. This is exactly $P(\mathbf{d},1)$ (up to the normalizing constant). The second job also contributes a factor of $\frac{\lambda_{\text{class}}}{\mu\mathbf{d}}$ and there is only one way of choosing the second job's class so that it shares all $\mathbf{d}$ servers with the first job, namely $c_2 = c_1$. Hence we find $P(\mathbf{d},2) = \frac{\lambda_{\text{class}}}{\mu\mathbf{d}} \cdot P(\mathbf{d},1)$. For example, when $k = 4$ and $\mathbf{d} = 2$ (see Figure 2), we find that $P(2,2) = \frac{\lambda_{\text{class}}}{2\mu} \cdot P(2,1)$.

Similarly, to find $P(\mathbf{d}+1,2)$, we first consider the contribution of the first job to the limiting probability. Again, the first job contributes a factor of

$$\binom{k}{\mathbf{d}}\frac{\lambda_{\text{class}}}{\mu\mathbf{d}} = P(\mathbf{d},1) \tag{7}$$

since class c_1 has arrival rate λ_{class}, $\mathbf{d}$ servers are busy working on this job, and there are $\binom{k}{\mathbf{d}}$ possible choices for the specific class c_1. The second job contributes a factor of

$$\frac{\lambda_{\text{class}}}{\mu(\mathbf{d}+1)} \cdot \left(\begin{array}{c}\text{\# ways to choose 2nd job}\\ \text{so it shares } \mathbf{d}-1 \text{ servers}\\ \text{with first job}\end{array}\right) = \frac{\lambda_{\text{class}}}{\mu(\mathbf{d}+1)} \cdot \binom{\mathbf{d}}{\mathbf{d}-1} \cdot \binom{k-\mathbf{d}}{1}, \tag{8}$$

where the $\binom{\mathbf{d}}{\mathbf{d}-1}$ term gives the number of ways that the second job can choose $\mathbf{d}-1$ servers in common with the first job and the $\binom{k-\mathbf{d}}{1}$ term gives the number of ways for the second job to choose one server that is different from all $\mathbf{d}$ of the first job's servers. Combining (7) and (8),

$$P(\mathbf{d}+1,2) = \frac{\lambda_{\text{class}}}{\mu(\mathbf{d}+1)} \cdot \binom{\mathbf{d}}{\mathbf{d}-1} \cdot \binom{k-\mathbf{d}}{1} \cdot P(\mathbf{d},1).$$

In the case where $k = 4$ and $\mathbf{d} = 2$, the graph in Figure 2 tells us that there are four ways in which the second job can choose servers such that it shares one server with the first job (that is, $\binom{\mathbf{d}}{\mathbf{d}-1} \cdot \binom{k-\mathbf{d}}{1} = \binom{2}{1} \cdot \binom{4-2}{1} = 4$). Thus the recurrence for $P(3,2)$ in the $k = 4$, $\mathbf{d} = 2$ system is

$$P(3,2) = \frac{\lambda_{\text{class}}}{3\mu} \cdot 4 \cdot P(2,1).$$

In general, when writing a recurrence for $P(i,m)$ we consider all possible values of y, the number of new servers busy working on the mth job. Equivalently, $i-y$ servers must be busy working on the first $m-1$ servers. Except in edge cases where there are already at least $k-\mathbf{d}+1$ servers working on the first $m-1$ jobs, the value of y can range from 0 to $\mathbf{d}$.

Given that $i-y$ servers are busy working on the first $m-1$ jobs, the contribution of the first $m-1$ jobs is $P(i-y,m-1)$. The mth job contributes

$$\frac{\lambda_{\text{class}}}{\mu i} \cdot \left(\begin{array}{c}\text{\# ways to choose } m\text{th job}\\ \text{so it shares } \mathbf{d}-y \text{ servers}\\ \text{with first } m-1 \text{ jobs}\end{array}\right) = \frac{\lambda_{\text{class}}}{\mu i} \cdot \binom{i-y}{\mathbf{d}-y} \cdot \binom{k-(i-y)}{y}$$

to $P(i,m)$, where the term $\binom{i-y}{\mathbf{d}-y}$ gives the number of ways to choose the $\mathbf{d}-y$ servers that the mth job shares with the first $m-1$ jobs from among the $i-y$ servers busy working on the first $m-1$ jobs, and the term $\binom{k-(i-y)}{y}$ gives the number of ways to choose y new servers from the remaining $k-(i-y)$ servers.

Finally, we condition on the number of new servers y to obtain the general form of $P(i,m)$ given in (4). $\square$

4.2.2 Finding Mean Response Time

Now that we have a form for $P(i,m)$, we can imagine finding the mean number in system $\mathrm{E}[N]$ by summing over all possible numbers of busy servers and all possible numbers of jobs in the system:

$$\mathrm{E}[N] = \sum_{i=\mathbf{d}}^{k}\sum_{m=1}^{\infty} mP(i,m)\cdot\mathcal{C},$$

where $\mathcal{C}$ is our normalizing constant. Unfortunately, computing these sums would require having an explicit form for $P(i,m)$, which is difficult to compute. Instead, we will find $\mathrm{E}[N]$ using generating functions.

We begin by rewriting our recurrences $P(i,m)$ in a form that eliminates the dependency on k. Starting with the expression given in (4), we substitute $\lambda_{\text{class}} = \frac{k\lambda}{\binom{k}{\mathbf{d}}}$ and rearrange the combinatorial terms to obtain:

$$P(i,m) = \frac{k\lambda}{\mu i}\sum_{y=0}^{\mathbf{d}} \frac{\binom{\mathbf{d}}{y}\binom{k-\mathbf{d}}{i-\mathbf{d}}}{\binom{k}{i-y}}\cdot P(i-y,m-1).$$

Our next step is to eliminate the $\binom{k}{i-y}$ term in the denominator. Let $Q(i,m) = \frac{1}{\binom{k}{i}}\cdot P(i,m)$. We then have

$$\begin{aligned}
Q(i,m)\cdot\binom{k}{i} &= P(i,m)\\
&= \frac{k\lambda}{\mu i}\sum_{y=0}^{\mathbf{d}} \frac{\binom{\mathbf{d}}{y}\binom{k-\mathbf{d}}{i-\mathbf{d}}}{\binom{k}{i-y}}\cdot P(i-y,m-1)\\
&= \frac{k\lambda}{\mu i}\sum_{y=0}^{\mathbf{d}} \frac{\binom{\mathbf{d}}{y}\binom{k-\mathbf{d}}{i-\mathbf{d}}}{\binom{k}{i-y}}\cdot \binom{k}{i-y}Q(i-y,m-1)\\
&= \frac{k\lambda}{\mu i}\cdot\binom{k-\mathbf{d}}{i-\mathbf{d}}\sum_{y=0}^{\mathbf{d}}\binom{\mathbf{d}}{y}Q(i-y,m-1).
\end{aligned}$$

Multiplying both sides by $\frac{i}{k}$, we get

$$Q(i,m)\cdot\binom{k-1}{i-1} = \frac{\lambda}{\mu}\cdot\binom{k-\mathbf{d}}{i-\mathbf{d}}\sum_{y=0}^{\mathbf{d}}\binom{\mathbf{d}}{y}Q(i-y,m-1).$$

Next, we will eliminate the $\frac{\lambda}{\mu}$ term from the recurrence. Let $R(i,m) = \left(\frac{\mu}{\lambda}\right)^m \cdot Q(i,m)$. Then we have

$$R(i,m) \cdot \binom{k-1}{i-1} = \binom{k-\mathbf{d}}{i-\mathbf{d}} \sum_{y=0}^{\mathbf{d}} \binom{\mathbf{d}}{y} R(i-y, m-1).$$

Finally, to eliminate the dependency on k, we let $S(i,m) = \binom{k-1}{\mathbf{d}-1}^m \cdot R(i,m)$ and obtain

$$\begin{aligned} S(i,m)\binom{k-1}{i-1} &= \binom{k-1}{\mathbf{d}-1}\binom{k-\mathbf{d}}{i-\mathbf{d}} \sum_{y=0}^{\mathbf{d}} \binom{\mathbf{d}}{y} S(i-y, m-1) \\ S(i,m) &= \binom{i-1}{\mathbf{d}-1} \sum_{y=0}^{\mathbf{d}} \binom{\mathbf{d}}{y} S(i-y, m-1). \end{aligned}$$

Note that $S(i,m)$ relates to our original recurrence $P(i,m)$ as follows:

$$P(i,m) = \frac{\binom{k}{i}\left(\frac{\lambda}{\mu}\right)^m}{\binom{k-1}{\mathbf{d}-1}^m} \cdot S(i,m). \tag{9}$$

We will now define a generating function for $S(i,m)$:

$$G_i(x) = \sum_{m=1}^{\infty} S(i,m)x^m.$$

Taking the derivative of this generating function, we obtain

$$\begin{aligned} G_i'(x) &= \sum_{m=1}^{\infty} m \cdot S(i,m)x^{m-1} \\ xG_i'(x) &= \sum_{m=1}^{\infty} m \cdot S(i,m)x^m \\ \sum_{i=\mathbf{d}}^{k} \binom{k}{i} xG_i'(x) &= \sum_{i=\mathbf{d}}^{k} \sum_{m=1}^{\infty} \binom{k}{i} m \cdot S(i,m)x^m, \end{aligned} \tag{10}$$

where the second line results from multiplying both sides of the equation by x and the third line results from multiplying both sides of the equation by $\binom{k}{i}$ and summing over all $\mathbf{d} \leq i \leq k$. Evaluating (10) at $x_0 = \frac{\lambda/\mu}{\binom{k-1}{\mathbf{d}-1}}$ we have

$$\sum_{i=\mathbf{d}}^{k} \binom{k}{i} x_0 G_i'(x_0) = \sum_{i=\mathbf{d}}^{k} \sum_{m=1}^{\infty} m \cdot P(i,m) = \frac{\mathrm{E}[N]}{\mathcal{C}},$$

which is exactly what we want, noting that we already know $\mathcal{C}$ from Theorem 2.

All we need to do is find $G_i'(x)$. Observe that if we evaluate $G_i(x)$ at $x_0 = \frac{\lambda/\mu}{\binom{k-1}{\mathbf{d}-1}}$ we get

$$\binom{k}{i} G_i(x_0) = \binom{k}{i} \sum_{m=1}^{\infty} S(i,m) x_0^m = \sum_{m=1}^{\infty} P(i,m) = \frac{\pi_i}{\mathcal{C}}, \tag{11}$$

where π_i is the limiting probability that i servers are busy. Furthermore, since the limiting probabilities have to sum to 1, we have the normalization equation

$$\frac{1}{\mathcal{C}} = 1 + \sum_{i=\mathbf{d}}^{k} \frac{\pi_i}{\mathcal{C}}. \tag{12}$$

We define the function $C(x)$:

$$C(x) = \prod_{i=\mathbf{d}}^{k} \left(1 - \binom{i-1}{\mathbf{d}-1} x\right).$$

Note that $C(x_0) = \mathcal{C}$ at $x_0 = \frac{\lambda/\mu}{\binom{k-1}{\mathbf{d}-1}}$.

Combining (12) and (11), we have

$$\frac{1}{C(x_0)} = 1 + \sum_{i=\mathbf{d}}^{k} \binom{k}{i} G_i(x_0). \tag{13}$$

Since $\frac{\lambda}{\mu}$ can range from 0 to 1, x_0 can take on any value from 0 to $\frac{1}{\binom{k-1}{\mathbf{d}-1}}$, so (13) holds for all $x \in (0, \frac{1}{\binom{k-1}{\mathbf{d}-1}})$. This allows us to differentiate both sides of (13), to get

$$\frac{d}{dx} \frac{1}{C(x)} = \sum_{i=\mathbf{d}}^{k} \binom{k}{i} G_i'(x)$$
$$x \frac{d}{dx} \frac{1}{C(x)} = \sum_{i=\mathbf{d}}^{k} \binom{k}{i} x G_i'(x). \tag{14}$$

Note that when evaluated at $x_0 = \frac{\lambda/\mu}{\binom{k-1}{\mathbf{d}-1}}$, the right-hand side of (14) is equal to $\frac{\mathrm{E}[N]}{\mathcal{C}}$.

So we have

$$\mathrm{E}[N] = C(x_0) \cdot x_0 \cdot \left(\frac{d}{dx} \frac{1}{C(x)} \right)\bigg|_{x=x_0} = \sum_{i=\mathbf{d}}^{k} \frac{\lambda}{\mu \frac{\binom{k-1}{\mathbf{d}-1}}{\binom{i-1}{\mathbf{d}-1}} - \lambda},$$

where the final equality results from taking the derivative of $\frac{1}{C(x)}$.

Finally, by Little's Law we have $\mathrm{E}[T] = \frac{\mathrm{E}[N]}{k\lambda}$, which gives us the form for $\mathrm{E}[T]$ given in (1). This completes the proof of Theorem 1.

4.3 Proof of Theorem 3

Theorem 3. *The Redundancy-*$\mathbf{d}$ *system is stable when* $\rho = \frac{\lambda}{\mu} < 1$.

Proof. Consider the system as the number of jobs $m \to \infty$. For any given number of busy servers $i < k$, the probability of increasing the number of busy servers when going from m to $m+1$ jobs is greater than $1/\binom{k}{\mathbf{d}} > 0$ and is independent of m. Hence as $m \to \infty$, the probability that all k servers are busy approaches 1 no slower than the c.d.f. of a geometric random variable with parameter $1/\binom{k}{\mathbf{d}}$. Thus as $m \to \infty$, $P(i, m) \to 0$ for all $i < k$, and so $\pi_m \to P(k, m)$. Looking at the recurrence for $P(k, m)$ given in (4), since $P(i, m) \to 0$ for all $i < k$, the tail terms of $P(k, m)$ are all of the form

$$\frac{\binom{k}{\mathbf{d}}\lambda_{\text{class}}}{k\mu} P(k, m-1) = \frac{\binom{k}{\mathbf{d}}\frac{k\lambda}{\binom{k}{\mathbf{d}}}}{k\mu} P(k, m-1) = \frac{\lambda}{\mu} P(k, m-1).$$

When $\lambda < \mu$, this term is less than 1 and so the $P(k, m)$'s form a geometric sequence. Hence the series $\sum_{m=0}^{\infty} \pi_m$ converges if and only if $\frac{\lambda}{\mu} < 1$. Since the series converges, there is some constant $\mathcal{C}$ such that the π_m's sum to 1. $\square$

5 Large System Limit Analysis

In Section 4 we derived exact expressions for mean response time in the Redundancy-$\mathbf{d}$ system for any specific k and $\mathbf{d}$ using a Markov chain approach. Even though the Markov chain approach gives us the full distribution of the number of jobs in the system, we cannot apply Distributional Little's Law to find the distribution of response time because jobs need not leave the system in the order in which they arrived. In this section we provide an alternative approach to analyzing the Redundancy-$\mathbf{d}$ system that yields a closed-form expression for the distribution of response time. Our result is exact in the limiting regime in which $k \to \infty$, under the assumption that the queues are *asymptotically independent*.

To understand what we mean by asymptotic independence, we first define a job's "non-redundant response time" on a server i to be the response time that the job would experience if it arrived to the Redundancy-$\mathbf{d}$ system and sent only one copy to a randomly chosen server i. The queues are $\mathbf{d}$-wise asymptotically independent if knowing a job's non-redundant response time on servers $i_1, \ldots, i_{\mathbf{d}-1}$ does not tell us anything about the job's non-redundant response time on server $i_{\mathbf{d}}$. Assumption 1 formalizes this notion of asymptotic independence.

Assumption 1. *In the Redundancy-d system, as* $k \to \infty$*, the queues are* $\mathbf{d}$*-wise asymptotically independent. That is,* $\Pr\{T_{i_{\mathbf{d}}} > t \mid T_{i_1}, \ldots, T_{i_{\mathbf{d}-1}}\} = \Pr\{T_{i_{\mathbf{d}}} > t\}$ *for all* $i_{\mathbf{d}} \neq i_1, \ldots, i_{\mathbf{d}-1}$*, where* T_i *is a job's non-redundant response time at server* i*.*

Theorem 6. *Under Assumption 1, as* $k \to \infty$*, the response time in the Redundancy-*$\mathbf{d}$ *system with* $\mathbf{d} > 1$ *has c.c.d.f.*

$$\Pr\{T > t\} = \bar{F}_T(t) = \left(\frac{1}{\rho + (1-\rho)e^{t\mu(\mathbf{d}-1)}}\right)^{\frac{\mathbf{d}}{\mathbf{d}-1}}, \tag{15}$$

where $\rho = \frac{\lambda}{\mu}$.

Conjecture 1. *Assumption 1 holds.*

Remark The analogue of Conjecture 1 has been proved in a wide range of settings: asymptotic independence of queues was shown under the JSQ-$\mathbf{d}$ policy in [17] for exponential service times, and extended to general service times in [5]. In [19], a similar result was shown for a variety of dispatching policies in a system with batch arrivals. Unfortunately, the proofs presented in the above work do not extend to the Redundancy-$\mathbf{d}$ system,[1] thus we consign proving Conjecture 1 to future work. In Section 5.2 we compare our analytical results to simulation and see that the results converge, supporting Conjecture 1.

We now turn to the proof of Theorem 6.

Proof. **[Theorem 6]** We consider a tagged arrival to the Redundancy-$\mathbf{d}$ system, which we assume without loss of generality arrived at time 0 to a system that is stationary. We want to find

$$\bar{F}_T(t) = \Pr\{\text{tagged arrival is not complete by time } t\}.$$

Denote by T_i the non-redundant response time of a job on server i, i.e., the time from when a job arrives at server i to when it would complete on server i if it had no other copies; note that T_i might be longer than response time T since T is the min of $T_1, \ldots, T_{\mathbf{d}}$. Throughout this section, T will always represent the response time in a Redundancy-$\mathbf{d}$ system, whereas T_i represents the non-redundant response time at server i.

We can express T in terms of T_i as follows:

$$\begin{aligned}\bar{F}_T(t) = \Pr\{T > t\} &= \Pr\{T_1 > t \;\&\; T_2 > t \;\&\; \cdots \;\&\; T_{\mathbf{d}} > t\} \\ &= \Pr\{T_1 > t\} \cdot \Pr\{T_2 > t\} \cdots \Pr\{T_{\mathbf{d}} > t\} \\ &= (\Pr\{T_i > t\})^{\mathbf{d}} \\ &= \bar{F}_{T_i}(t)^{\mathbf{d}}, \end{aligned} \tag{16}$$

where the second line is due to the asymptotic independence assumption. Thus in order to find $\bar{F}_T(t)$, we need to understand $\bar{F}_{T_i}(t)$.

To understand $\bar{F}_{T_i}(t)$, observe that there are two ways in which a tagged arrival could have not completed service at server i by time t (assuming the tagged job has no other copies). First, the tagged job could have size larger than t at server i. Second, even if the tagged job has size $S_i < t$, it will not complete at server i by time t if it does not enter service at server i by time $t - S_i$, that

[1]We have discussed the asymptotic independence conjecture with several of the authors of [5], all of whom agree that while the conjecture almost certainly is correct, the existing proof techniques will not apply to the Redundancy-$\mathbf{d}$ system.

is, if its non-redundant time in queue at server i, T_i^Q, exceeds $t - S_i$. We thus have

$$\begin{aligned}
\bar{F}_{T_i}(t) &= \Pr\{T_i > t\} \\
&= \Pr\{S_i > t\} + \Pr\{0 < S_i < t \ \wedge\ T_i^Q > t - S_i\} \\
&= \bar{F}_S(t) + \int_0^t f_S(x) \cdot \bar{F}_{T_i^Q}(t-x)dx \\
&= e^{-\mu t} + \int_0^t \mu e^{-\mu x} \cdot \bar{F}_{T_i^Q}(t-x)dx \\
&= e^{-\mu t} + \int_0^t \mu e^{-\mu(t-y)} \cdot \bar{F}_{T_i^Q}(y)dy,
\end{aligned} \tag{17}$$

where the integral is due to conditioning on the value of S_i. We have now expressed $\bar{F}_{T_i}(t)$ in terms of $\bar{F}_{T_i^Q}(t)$.

Next we need to understand $\bar{F}_{T_i^Q}(t)$, the probability that the tagged job has not entered service at server i by time t (assuming the tagged job has no other copies). To do this, we look back in time to the most recent arrival to server i before the tagged job arrived. Call this most recent arrival job A. Suppose job A arrived at time $t - Y < 0$. The tagged job will not enter service by time t if and only if either

1. Job A cannot have entered service at server i by time t because there is still some other job ahead of it. This is equivalent to saying that for job A, $T_i^Q > Y$, recalling that T_i^Q is the time that job A would spend in the queue at server i if it had no other copies.

2. Job A is in service at server i at time t. That is, job A has not departed from server i or from any of its other $\mathbf{d} - 1$ servers by time t.

We thus have

$$\begin{aligned}
\Pr\left\{\begin{array}{c}\text{tagged job not}\\ \text{in service by}\\ \text{time } t\end{array}\right\} &= \Pr\left\{\begin{array}{c}\text{job } A \text{ cannot have}\\ \text{entered service at}\\ \text{server } i \text{ by time } t\end{array}\right\} + \Pr\left\{\begin{array}{c}\text{job } A \text{ is in service}\\ \text{at server } i \text{ by time } t\\ \text{but has not departed}\\ \text{any server by time } t\end{array}\right\} \\
&= \Pr\left\{T_i^Q > Y\right\} + \Pr\left\{T_i^Q < Y \ \wedge\ T > Y\right\} \\
&= \bar{F}_{T_i^Q}(Y) + \Pr\left\{T_i^Q < Y \ \wedge\ T_1 > Y \wedge \cdots \wedge T_{\mathbf{d}} > Y\right\} \\
&= \bar{F}_{T_i^Q}(Y) + \Pr\left\{T_i^Q < Y \wedge T_i > Y\right\} \cdot \bar{F}_{T_i}(Y)^{\mathbf{d}-1} \\
&= \bar{F}_{T_i^Q}(Y) + (\bar{F}_{T_i}(Y) - \bar{F}_{T_i^Q}(Y))\bar{F}_{T_i}(Y)^{\mathbf{d}-1},
\end{aligned}$$

where again we assume that the $\mathbf{d}$ queues are independent.

Now to find $\bar{F}_{T_i^Q}(t)$, we integrate over all possible values of Y such that job A could have arrived at time $t - Y$, noting that the interarrival times to a particular queue are exponentially distributed with rate $\lambda\mathbf{d}$:

$$\bar{F}_{T_i^Q}(t) = \int_t^\infty \lambda\mathbf{d}e^{\lambda\mathbf{d}(t-y)} \left(\bar{F}_{T_i^Q}(y) + (\bar{F}_{T_i}(y) - \bar{F}_{T_i^Q}(y))\bar{F}_{T_i}(y)^{\mathbf{d}-1}\right) dy.$$

Note that $\bar{F}_{T_i}(t)$ and $\bar{F}_{T_i^Q}(t)$ are defined recursively in terms of each other.

We thus have a system of two differential equations:

$$\bar{F}_{T_i}(t) = e^{-\mu t} + \int_0^t \mu e^{-\mu(t-y)} \cdot \bar{F}_{T_i^Q}(y)dy \tag{18}$$

$$\bar{F}_{T_i^Q}(t) = \int_t^\infty \lambda\mathbf{d}e^{\lambda\mathbf{d}(t-y)} \left(\bar{F}_{T_i^Q}(y) + (\bar{F}_{T_i}(y) - \bar{F}_{T_i^Q}(y))\bar{F}_{T_i}(y)^{\mathbf{d}-1}\right) dy. \tag{19}$$

To solve the system, we begin by taking the derivative of (18), using the general Leibniz's rule to differentiate the function of two variables under the integral:

$$\begin{aligned}
\bar{F}'_{T_i}(t) &= -\mu e^{-\mu t} + \int_{y=0}^t \frac{d}{dt}\left(\mu e^{-\mu(t-y)}\bar{F}_{T_i^Q}(y)\right) dy + \mu e^{-\mu(t-t)}\bar{F}_{T_i^Q}(t) \cdot \frac{d}{dt}t - \mu e^{-\mu(t-0)}\bar{F}_{T_i^Q}(0) \cdot \frac{d}{dt}0 \\
&= -\mu e^{-\mu t} + \int_0^t -\mu^2 e^{-\mu(t-y)}\bar{F}_{T_i^Q}(y)dy + \mu\bar{F}_{T_i^Q}(t) \\
&= -\mu\left(e^{-\mu t} + \int_0^t \mu e^{-\mu(t-y)}\bar{F}_{T_i^Q}(y)dy - \bar{F}_{T_i^Q}(t)\right) \\
&= \mu\left(\bar{F}_{T_i^Q}(t) - \bar{F}_{T_i}(t)\right),
\end{aligned} \tag{20}$$

where the last line results from substituting (18). Taking the derivative of (19) in a similar manner, we find

$$\begin{aligned}
\bar{F}'_{T_i^Q}(t) &= \lambda\mathbf{d}\bar{F}_{T_i}(t)^{\mathbf{d}-1}\left(\bar{F}_{T_i^Q}(t) - \bar{F}_{T_i}(t)\right) \\
&= \frac{\lambda\mathbf{d}}{\mu}\bar{F}_{T_i}(t)^{\mathbf{d}-1} \cdot \bar{F}'_{T_i}(t),
\end{aligned} \tag{21}$$

where the last line results from substituting (20). Taking the derivative of (20), we find

$$\begin{aligned}
\bar{F}''_{T_i}(t) &= \mu\left(\bar{F}'_{T_i^Q}(t) - \bar{F}'_{T_i}(t)\right) \\
&= \lambda\mathbf{d}\bar{F}_{T_i}(t)^{\mathbf{d}-1} \cdot \bar{F}'_{T_i}(t) - \mu\bar{F}'_{T_i}(t),
\end{aligned} \tag{22}$$

where the last line results from substituting (21). Integrating (22), we get

$$\bar{F}'_{T_i}(t) = \lambda\bar{F}_{T_i}(t)^{\mathbf{d}} - \mu\bar{F}_{T_i}(t).$$

Now we have a single differential equation for $\bar{F}_{T_i}(t)$, which we solve to get

$$\bar{F}_{T_i}(t) = \left(\frac{\mu}{\lambda + \alpha e^{t\mu(d-1)}}\right)^{\frac{1}{\mathbf{d}-1}},$$

where α is a constant. Note that solving this differential equation is the only place where we needed $\mathbf{d} > 1$. We know that $\bar{F}_{T_i}(0) = 1$, so we can solve for α, yielding $\alpha = \mu - \lambda$. So we have

$$\bar{F}_{T_i}(t) = \left(\frac{\mu}{\lambda + (\mu - \lambda)e^{t\mu(\mathbf{d}-1)}}\right)^{\frac{1}{\mathbf{d}-1}}.$$

Finally, we need $\bar{F}_T(t)$, which from (16) is

$$\begin{aligned}\bar{F}_T(t) &= \bar{F}_{T_i}(t)^{\mathbf{d}} \\ &= \left(\frac{\mu}{\lambda + (\mu - \lambda)e^{t\mu(\mathbf{d}-1)}}\right)^{\frac{\mathbf{d}}{\mathbf{d}-1}}.\end{aligned}$$

An alternative way of writing this is

$$\bar{F}_T(t) = \left(\frac{1}{\rho + (1 - \rho)e^{t\mu(\mathbf{d}-1)}}\right)^{\frac{\mathbf{d}}{\mathbf{d}-1}}$$

as desired. □

Once we have the c.c.d.f. of response time in the Redundancy-$\mathbf{d}$ system, we can integrate this over all values of t to find mean response time $\mathrm{E}[T]$. In Theorem 7, we see that $\mathrm{E}[T]$ can be expressed in terms of the hypergeometric function. When $\mathbf{d} = 2$, $\mathrm{E}[T]$ has a simple closed form.

Theorem 7. *The mean response time in the infinite-server Redundancy-*$\mathbf{d}$ *system is*

$$\mathbb{E}[T] = \frac{{}_2F_1(1,\ 1;\ 1 + \frac{\mathbf{d}}{\mathbf{d}-1};\ \frac{-\rho}{1-\rho})}{\mu\mathbf{d}(\rho - 1)}, \tag{23}$$

where

$${}_2F_1(a, b; c; z) = \sum_{n=0}^{\infty} \frac{a^{(n)} b^{(n)}}{c^{(n)}} \frac{z^n}{n!}$$

is the hypergeometric function and

$$x^{(n)} = \begin{cases} 1 & n = 0 \\ x(x+1)\cdots(x+n-1) & n > 0 \end{cases}$$

is the rising Pochammer symbol.

In the case $\mathbf{d} = 2$*, this is equivalent to*

$$\mathbb{E}[T_{d=2}] = \frac{\mu \ln\left(\frac{\mu}{\mu-\lambda}\right) - \lambda}{\lambda^2}. \tag{24}$$

Proof. This follows directly from Theorem 6 by integrating $\bar{F}_T(t)$ given in (16) over all values of t. □

It is worth comparing the expression in (24) to the mean response time under JSQ-2, in which each arriving job polls 2 servers and joins only that queue which is the shorter of the two. From [13, 17] it is known that when $\mu = 1$ and $\lambda \to 1$ the mean response time under JSQ-2 is given by

$$\mathrm{E}[T] = \frac{\ln\left(\frac{1}{1-\lambda}\right)}{\lambda \ln(2\lambda)} + O(1). \tag{25}$$

Thus for $\mathbf{d} = 1$ as $\lambda \to \mu = 1$ the mean response times in both systems have the term $\ln(\frac{1}{1-\lambda})$ in common.

5.1 Insights

Theorem 6 tells us the distribution of response time in the infinite-server Redundancy-$\mathbf{d}$ system. In this section, we discuss the characteristics of system behavior that follow from the form of this distribution.

Theorem 8. *The response time in the infinite-server Redundancy-*$\mathbf{d}$ *system has increasing failure rate.*

Proof. We first find the failure rate

$$\begin{aligned} r_T(t) &= \frac{f_T(t)}{\bar{F}_T(t)} \\ &= \frac{(1-\rho)\mu\mathbf{d}e^{t\mu(\mathbf{d}-1)}}{(1-\rho)e^{t\mu(\mathbf{d}-1)} + \rho}. \end{aligned}$$

Now we find the derivative of $r_T(t)$ as follows:

$$r'_T(t) = \frac{\mathbf{d}(\mathbf{d}-1)\rho(1-\rho)\mu^2 e^{t\mu(\mathbf{d}-1)}}{(\rho + (1-\rho)e^{t\mu(\mathbf{d}-1)})^2},$$

which is positive since the denominator is positive and all terms in the numerator are positive. Hence the response time distribution has increasing failure rate. □

The intuition behind this result is that as time passes, a job is likely to be in service at more and more servers, so its probability of completing ("failing") increases.

Lemma 1 tells us that although the response time distribution has increasing failure rate, as $t \to \infty$, the failure rate approaches $\mu\mathbf{d}$. This is because once a job has been in the system for a very long time, it is in service at all $\mathbf{d}$ of its servers. At this point, the remaining time to completion is simply the minimum of $\mathbf{d}$ exponentials with rate μ, which is an exponential with rate $\mu\mathbf{d}$.

Lemma 1. *As $t \to \infty$, the failure rate of the response time distribution in the Redundancy-*$\mathbf{d}$ *system approaches* $\mu\mathbf{d}$.

Proof. From Theorem 8, we have that

$$r_T(t) = \frac{(1-\rho)\mu \mathbf{d} e^{t\mu(\mathbf{d}-1)}}{(1-\rho)e^{t\mu(\mathbf{d}-1)} + \rho}.$$

Taking the limiting as $t \to \infty$, we find

$$\begin{aligned}
\lim_{t\to\infty} r_T(t) &= \lim_{t\to\infty} \frac{(1-\rho)\mu \mathbf{d} e^{t\mu(\mathbf{d}-1)}}{(1-\rho)e^{t\mu(\mathbf{d}-1)} + \rho} \\
&= \lim_{t\to\infty} \frac{\mu \mathbf{d}}{1 + \frac{\rho}{(1-\rho)e^{t\mu(\mathbf{d}-1)}}} \\
&= \mu \mathbf{d}.
\end{aligned}$$

□

Lemma 1 addresses what happens to a job's remaining response time given that it has been in the system for a long time. In Lemma 2, we look at the effect of $\mathbf{d}$ on the response time distribution.

Lemma 2. *As* $\mathbf{d} \to \infty$*, mean response time scales as* $\frac{1}{\mathbf{d}}$*.*

Proof. Theorem 6 tells us that

$$\bar{F}_T(t) = \left(\frac{1}{\rho + (1-\rho)e^{t\mu(\mathbf{d}-1)}}\right)^{\frac{\mathbf{d}}{\mathbf{d}-1}}.$$

As $\mathbf{d} \to \infty$, the exponent approaches 1, so we have

$$\bar{F}_T(t) \to \frac{1}{\rho + (1-\rho)e^{t\mu(\mathbf{d}-1)}}.$$

Integrating over all t to find mean response time, we find

$$\mathrm{E}[T] = \frac{\ln(\frac{1}{1-\rho})}{\mathbf{d}\mu\rho}.$$

□

Lemma 2 tells us that as $\mathbf{d}$ becomes large, we see a *diminishing marginal improvement* from further increasing $\mathbf{d}$. This makes sense: when a job is only running on one server, adding an additional server can make a big difference. But when a job is already running on many servers, one extra server does not add very much processing power relative to what the job already is experiencing. This is important because it suggests that the biggest improvement in response time will come from moving from $\mathbf{d} = 1$ to $\mathbf{d} = 2$, that is, creating just one extra copy of each job. We further explore this result in Section 6.

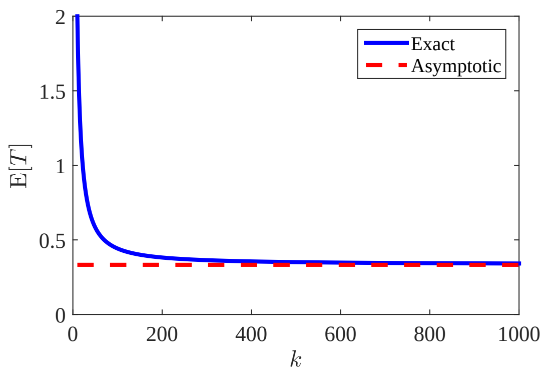


Figure 3: Convergence of the mean response time $\mathrm{E}[T]$ in the finite k server system (blue solid line) to that in the infinite system (red dashed line) when $\rho = 0.95$ and $\mathbf{d} = 10$. As k increases, $\mathrm{E}[T]$ in the finite system drops very steeply to meet that in the infinite system.

		ρ							
		0.2	**0.5**	**0.9**	**0.95**	**0.2**	**0.5**	**0.9**	**0.95**
d	**2**	3	7	41	73	10	31	192	346
	4	7	18	105	190	23	78	496	904
	6	10	28	168	305	36	125	794	1450
	10	17	49	293	534	63	216	1387	2538

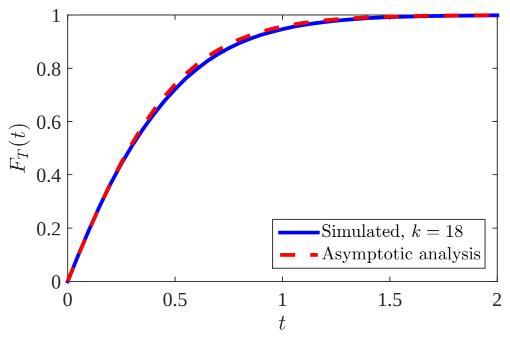
Table 1: Number of servers k at which the mean response time in the finite k server is within 5% (left) and 1% (right) of the asymptotic mean response time.

5.2 Convergence

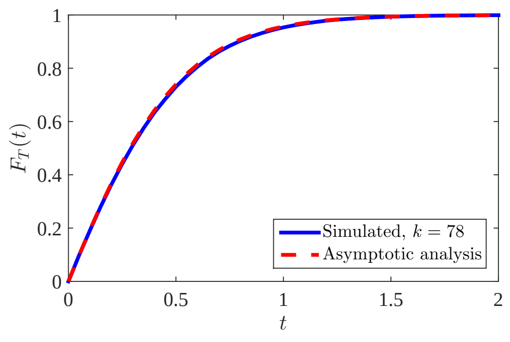
We now turn to the question of convergence: how high does k have to be for the asymptotic analysis to provide a reasonable approximation for a finite k-server system?

In Figure 3 we consider the convergence of the mean response time in a finite k-server system to that in the infinite system in the case of $\rho = 0.95$ and $\mathbf{d} = 10$. We see that when k is very low, the mean response time given by our asymptotic analysis is up to a factor of 5 smaller than the exact mean response time given by our analysis in Section 4. However as k increases mean response time drops very quickly and ultimately converges to the asymptotic result. This supports the asymptotic independence assumption we make when proving Theorem 6.

Table 1 shows convergence results for a broader range of parameters. We show the number of servers k required for the mean response time in the finite system to be within 5% (left) and within 1% (right) of that in the infinite system for different values of ρ and $\mathbf{d}$. We consider $\mathbf{d} = 2, 4, 6,$ and 10; as we will see in Section 6, higher values of $\mathbf{d}$ do not yield appreciable response time improvements.



(a) $k = 18$



(b) $k = 78$

Figure 4: Convergence of the full distribution in a system with $\mathbf{d} = 4$, $\lambda = 0.5$, and (a) $k = 18$ and (b) $k = 78$. The dashed red line shows the c.d.f. of response time in the limiting system (Theorem 6, Section 5) and the solid blue shows the simulated c.d.f. in the finite system (confidence intervals are within the line).

From Table 1 we see that the number of servers required for convergence increases in both ρ and $\mathbf{d}$. When load is low, only tens of servers are required for convergence at all values of $\mathbf{d}$ considered. Even at very high load ($\rho = 0.95$) and $\mathbf{d} = 10$, about 530 servers are sufficient for convergence within 5% and about 2500 servers are sufficient for convergence within 1%. This indicates that mean response time in the limiting system approximates that in the finite system very well for system sizes of practical interest. For example, typical data centers consist of hundreds or thousands of servers.

Thus far we have only considered convergence of the mean response time; we now turn to convergence of the full response time distribution. Since our exact analysis of the finite system only gives mean response time, we use simulation to compare the response time distribution in the finite k-server system with our asymptotically exact expression. We consider one cell in Table 1 – the case of $\mathbf{d} = 4$, $\rho = 0.5$. As shown in Table 1, 18 servers (respectively, 78 servers) suffice for convergence in the mean to 5% (respectively, 1%). Figure 4 shows convergence of the full response time distribution for (a) $k = 18$ servers, and (b) $k = 78$ servers, where the biggest difference between the empirical and analytical c.d.f.s is only 0.002. While we show only one value of $\mathbf{d}$ and ρ here, similar results hold for all other parameter choices we tested; the values of k corresponding to convergence of the mean to within 1% in Table 1 are typically also high enough for the response time c.d.f. in the finite system to appear virtually the same as that in the infinite system. This indicates that the distributional results obtained for the limiting system also can be used to understand the behavior of finite systems.

6 Power of d Choices

In this section we study the effect of increasing $\mathbf{d}$ on response time in the Redundancy-$\mathbf{d}$ system. We assume that k is large enough to allow us to leverage our asymptotic analysis from Section 5.

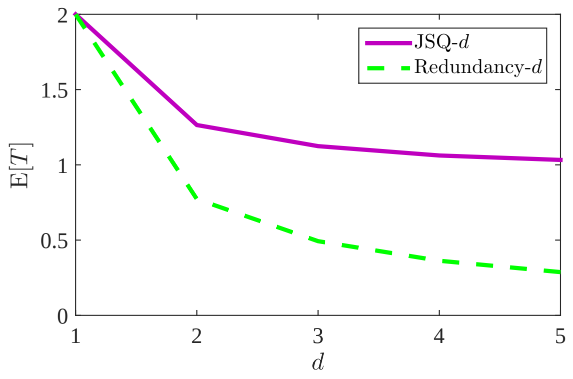


Figure 5: Comparing mean response time $\mathrm{E}[T]$ under Redundancy-$\mathbf{d}$ (dashed green line) and JSQ-$\mathbf{d}$ (solid purple line) as $\mathbf{d}$ increases when load is $\rho = 0.5$. Under both policies as $\mathbf{d}$ increases $\mathrm{E}[T]$ decreases; this improvement is much greater under Redundancy-$\mathbf{d}$. For JSQ-$\mathbf{d}$ (simulated), 95% confidence intervals are within the line.

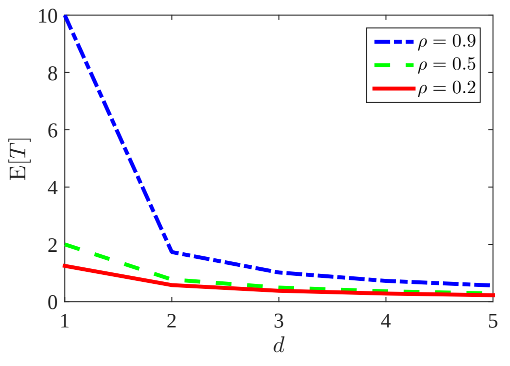
Throughout this section we assume the service rate at every server is $\mu = 1$.

Figure 5 compares mean response time as a function of $\mathbf{d}$ under Redundancy-$\mathbf{d}$ to that under the well-studied Join-the-Shortest-Queue-$\mathbf{d}$ (JSQ-$\mathbf{d}$, [13, 17]) dispatching policy when the system load, defined as $\rho = \frac{\lambda}{\mu}$, is $\rho = 0.5$. Under JSQ-$\mathbf{d}$, each arrival polls $\mathbf{d}$ servers chosen uniformly at random and joins the queue at the server with the fewest jobs in the queue. Note that jobs only join *one* queue under JSQ-$\mathbf{d}$; there is no redundancy. The Redundancy-$\mathbf{d}$ results are from our asymptotic analysis (Section 5); JSQ-$\mathbf{d}$ is simulated with $k = 1000$.

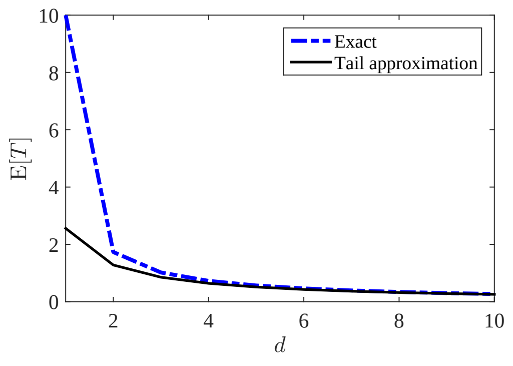
Like under JSQ-$\mathbf{d}$, we see that under Redundancy-$\mathbf{d}$ increasing $\mathbf{d}$ yields a substantial response time improvement relative to $\mathbf{d} = 1$ (no redundancy): both JSQ-$\mathbf{d}$ and Redundancy-$\mathbf{d}$ take advantage of queue length variability by allowing a job to wait in the shortest of $\mathbf{d}$ queues. But redundancy provides an additional benefit as well - the same job can be in service at multiple servers at the same time, in which case it experiences the minimum service time across all of these servers. This allows Redundancy-$\mathbf{d}$ to provide much lower response times than JSQ-$\mathbf{d}$.

Mean response time in the Redundancy-$\mathbf{d}$ system exhibits these same trends under other loads: Figure 6(a) shows mean response time in the Redundancy-$\mathbf{d}$ system as a function of $\mathbf{d}$ for low, medium, and high load (again we assume that k is large and thus show results from our asymptotic analysis). At all loads, as $\mathbf{d}$ increases, mean response time decreases, with this benefit being greatest under higher loads. When load is low, queueing times are low so the primary benefit of redundancy comes from a job receiving the minimum service time on $\mathbf{d}$ servers. When load is high, queueing times are typically higher so there is more opportunity to reduce response time by reducing queueing time as well as service time.

At all loads, the most significant improvement occurs between $\mathbf{d} = 1$ and $\mathbf{d} = 2$. This improvement ranges from a factor of 2 at $\rho = 0.2$ to a factor of 6 at $\rho = 0.9$. As $\mathbf{d}$ grows large, Lemma 2 tells us that mean response time scales as $\frac{1}{\mathbf{d}}$. This is shown in Figure 6(b), which compares our analytical result for $\mathrm{E}[T]$ to the tail form given in Lemma 2 when $\rho = 0.9$. We see



(a) Mean improvement



(b) $\frac{1}{\mathbf{d}}$ shape

Figure 6: (a) Mean response time $\mathrm{E}[T]$ in the Redundancy-$\mathbf{d}$ system as a function of $\mathbf{d}$ under low ($\rho = 0.2$, solid red line), medium ($\rho = 0.5$, dashed green line), and high ($\rho = 0.9$, dot-dashed blue line) load. At all loads increasing $\mathbf{d}$ reduces $\mathrm{E}[T]$. The improvement in $\mathrm{E}[T]$ is greatest at high load. (b) As $\mathbf{d}$ grows large, $\mathrm{E}[T]$ scales in proportion to $\frac{1}{\mathbf{d}}$, in accordance with Lemma 2.

that $\mathrm{E}[T]$ converges to the predicted tail shape relatively quickly; by $\mathbf{d} = 6$ the lines are nearly indistinguishable.

Thus far we have discussed only the mean response time; however our asymptotic analysis provides the full response time distribution. Figure 7 shows the c.d.f. of response time in the Redundancy-$\mathbf{d}$ system with $\mathbf{d} = 2$ at low, medium, high, and very high load. When load is high, not only is $F_T(t) = \Pr\{T < t\}$ lower, but the shape of the c.d.f. actually is *concave* at low values of t. This concavity is due to the probability of queueing: when load is high, an arrival is likely to experience a non-zero queueing time at all $\mathbf{d}$ of its servers. Thus the probability that the job completes service by time t does not increase substantially until t is sufficiently high, making it more likely that the job has entered service at one or more servers. In contrast, when load is low, an arrival is likely to begin service immediately on at least one server, so its probability of completing service by time t resembles the probability that the service time on a single server is less than t.

In Figure 8 we look at the c.d.f. of response time at high load ($\rho = 0.9$) as $\mathbf{d}$ increases from 2 to 6. When $\mathbf{d} = 2$, the c.d.f. is concave at low t (this is the same curve as shown in Figure 7). As $\mathbf{d}$ increases, the c.d.f. is still concave at low t but this is much less pronounced. At high $\mathbf{d}$, the c.d.f. approaches that of an exponential distribution because sending more copies means that a job is more likely to enter service immediately on at least one server. The concavity has disappeared, illustrating the dramatic effect redundancy can have on queueing time, and thus on system time. Looking at the tail of response time, we see from Figure 8 that $T_{95} = 3.58$ when $\mathbf{d} = 2$; this is $8\times$ smaller than T_{95} when $\mathbf{d} = 1$ (which corresponds to an M/M/1 system).

6.1 Fractional d

In Section 6 we saw that the largest improvement in mean response time occurred between $\mathbf{d} = 1$ (no redundancy) and $\mathbf{d} = 2$. Given the magnitude of this gap, we now explore the response time

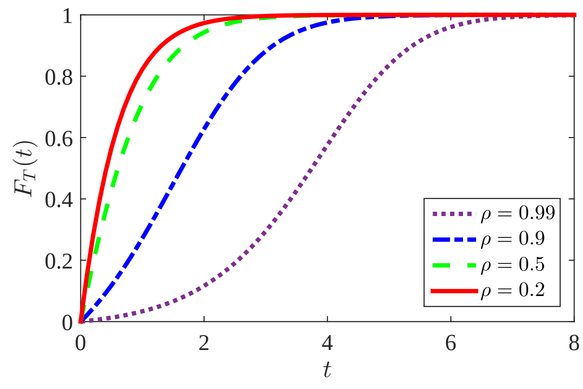


Figure 7: $\Pr\{T \leq t\}$ in the Redundancy-$\mathbf{d}$ system when $\mathbf{d} = 2$ under four different loads.

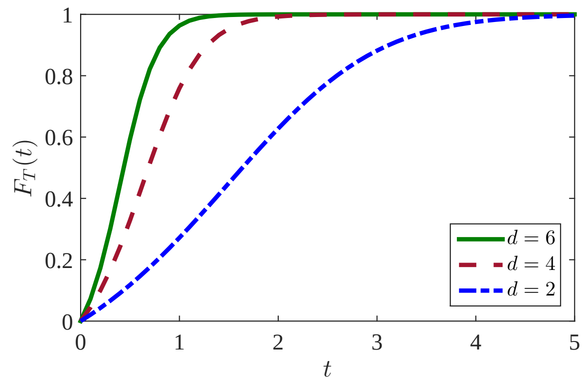


Figure 8: $\Pr\{T \leq t\}$ in the Redundancy-$\mathbf{d}$ system when $\rho = 0.9$ and $\mathbf{d} = 2$, 4, or 6.

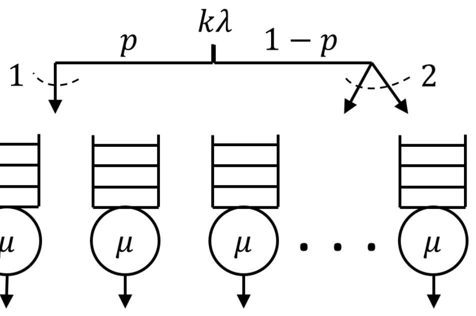


Figure 9: The fractional Redundancy-d system. With probability p an arriving job sends a request to a single server chosen uniformly at random, and with probability $1 - p$ an arriving job sends redundant requests to two servers chosen uniformly at random.

benefits offered by sending on average fewer than two copies of each job.

We define a fractional Redundancy-$\mathbf{d}$ system as shown in Figure 9. When a job arrives to the system, with probability p it is non-redundant and joins the queue at a single server chosen uniformly at random. With probability $1-p$ the job joins the queue at two servers chosen uniformly at random. In this system, we define $\mathbf{d}$ to be the weighted average number of copies sent per job:

$$\mathbf{d} = p \cdot 1 + (1-p) \cdot 2 = 2 - p.$$

To analyze mean response time in the fractional Redundancy-$\mathbf{d}$ system, we follow the Markov chain approach presented in Section 4, which extends easily to the fractional $\mathbf{d}$ case (unfortunately the asymptotic analysis in Section 5 does not extend to fractional $\mathbf{d}$). We obtain an exact closed-form expression for $\mathrm{E}[T]$, given in Theorem 9. The derivation of $\mathrm{E}[T]$ in the fractional Redundancy-$\mathbf{d}$ system follows the same approach as in Section 4, hence we omit the proof.

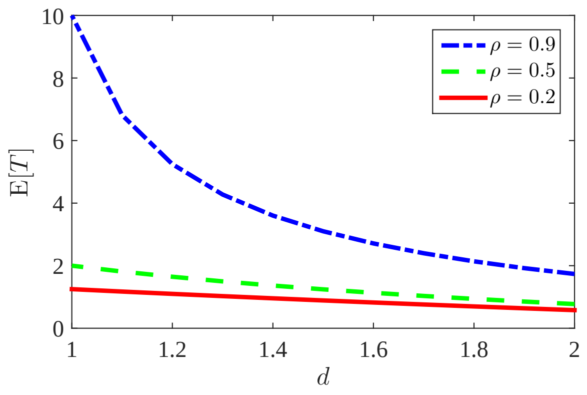


Figure 10: Mean response time $\mathrm{E}[T]$ as a function of $\mathbf{d}$ in the fractional Redundancy-$\mathbf{d}$ system under low ($\rho = 0.2$, solid red line), medium ($\rho = 0.5$, dashed green line), and high ($\rho = 0.9$, dot-dashed blue line) load.

Theorem 9. *The mean response time in the fractional Redundancy-$\mathbf{d}$ system is*

$$\frac{1}{k}\sum_{i=1}^{k}\frac{(i-1)+(k-i)p}{(k-1)\mu-((i-1)+(k-i)p)\lambda}. \tag{26}$$

Figure 10 shows mean response time $\mathrm{E}[T]$ as a function of $\mathbf{d}$ for $1 \le \mathbf{d} \le 2$ for low, medium, and high load when $k = 1000$. Note that Figure 10 is the same setting as Figure 6, but zooms in on the range from $\mathbf{d} = 1$ to $\mathbf{d} = 2$. As $\mathbf{d}$ increases, mean response time decreases convexly; introducing even a small amount of redundancy to the system provides a substantial improvement. This is particularly pronounced at high load; at $\rho = 0.9$ setting $\mathbf{d} = 1.5$ (i.e., half of the jobs are non-redundant) corresponds to a response time improvement of 69% relative to having no redundancy. Even at low load, $\mathrm{E}[T]$ is 29% lower when $\mathbf{d} = 1.5$ than when $\mathbf{d} = 1$.

Once again, very little redundancy is required to achieve significant performance gains. This result is encouraging for systems where there may be costs to redundancy, because it suggests that one can achieve response time benefits with only a limited amount of redundancy.

6.2 Non-Exponential Service Times

Thus far, we have assumed that service times are exponentially distributed. This assumption was necessary to obtain the closed-form results for mean response time given in Theorem 1 and for the distribution of response time given in Theorem 6. However, in real systems service times may not be exponential. For example, in computer systems network congestion can cause web query round trip times to be highly variable [18]. In this section we use our differential equations approach from Section 5 to study, numerically, what happens when service times are more or less variable than an exponential.

Returning to Section 5, we see that our argument allows us to write equations (17) and (19) regardless of the particular service time distribution S. In the case where $S \sim \mathrm{Exp}(\mu)$, we are able

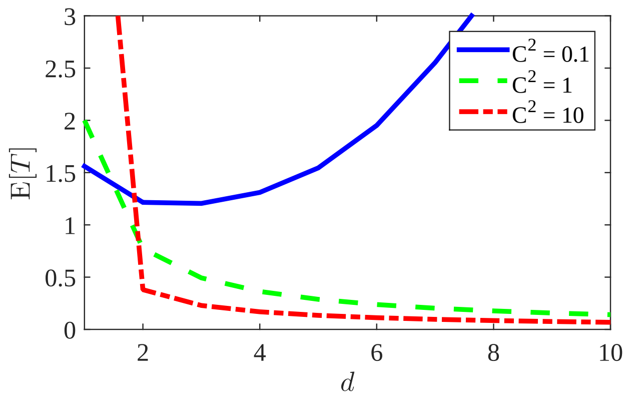


Figure 11: Mean response time as a function of $\mathbf{d}$ in the Redundancy-$\mathbf{d}$ system when service times are more or less variable than an exponential. Here $\lambda = 0.5$ and we assume k is large. Lines shown include $S \sim H_2$ with $C^2 = 10$ (dot-dashed red line), $S \sim \text{Exp}$ (dashed green line), and $S \sim \text{Erlang}$ with $C^2 = 0.1$ (solid blue line). For all distributions $\mathrm{E}[S] = 1$.

to solve the system in closed form. For non-exponential service times, while we are unable to find a closed-form solution, we can solve our differential equations numerically.

Figure 11 shows mean response time as a function of $\mathbf{d}$ when service times are more and less highly variable than an exponential and $\lambda = 0.5$. When service times are highly variable, increasing the value of $\mathbf{d}$ reduces mean response time even more than when service times are exponential. For example, when $C^2 = 10$, mean response time decreases by a factor of 17 (compared to a factor of 2.6 for exponentially distributed service times). The improvement is bigger under more highly variable service times for two reasons. First, when $\mathbf{d} = 1$ (i.e., there is no redundancy) queueing times can be extremely high when service times are highly variable, and waiting in multiple queues helps jobs to avoid waiting behind very long jobs. Second, a job that runs on multiple servers benefits from seeing the minimum service time across servers, and taking the minimum of multiple service times leads to a larger improvement when the service times are more variable.

The trend is very different when service times have low variability. Unlike for higher variability job size distributions, going from $\mathbf{d} = 1$ to $\mathbf{d} = 2$ yields only a small improvement in mean response time, and as $\mathbf{d}$ becomes higher mean response time actually *increases*. This is because queueing times are already quite low when service times have low variability, and rather than benefiting from running on multiple servers, adding multiple copies of similarly-sized jobs congests the system. Thus the "power-of-$\mathbf{d}$" depends crucially not only on load, but also on service time variability.

7 Conclusion

Redundancy is an important new technique used in queueing systems to reduce response time. A natural dispatching policy for systems with redundancy is to create $\mathbf{d}$ copies of each job, sending them to $\mathbf{d}$ different servers chosen uniformly at random. In this paper we provide the first exact analysis of response time in this Redundancy-$\mathbf{d}$ system.

We first model the system as a Markov chain that tracks a very detailed state space. While the limiting distribution on this state space follows from [8], aggregating the state space to get the distribution of the number of jobs in the system is combinatorially challenging. Our key insight is that we can derive π_m, the probability that there are m jobs in the system, by further conditioning on the probability that there are m jobs in the system and i servers busy working on these jobs. Expressing π_m in this manner yields a recursive structure that we leverage to find the distribution of the number of jobs in the system and the mean response time in systems with any number of servers k and any number of copies per job $\mathbf{d}$.

In our second analytical approach, we consider the system in the limit as the number of servers approaches infinity. In such a setting we are able to capture the Redundancy-$\mathbf{d}$ system's behavior via a system of differential equations that track the amount of work seen by a tagged arrival. We use these differential equations to derive asymptotic expressions for the distribution of response time that are exact under an independence assumption.

Our analysis allows us to answer questions about the benefits of redundancy that have important implications for real systems. For example, in Section 6 we saw that being redundant in only two places is enough to give most of the response time benefit of redundancy. For organ transplant patients, this suggests that multiple listing in only a small number of regions suffices. Furthermore, in Section 6.1 we saw that much of the response time benefit of redundancy can be achieved when only a small fraction of jobs are redundant. Many patients may be unable to multiple list at all because they cannot travel to alternative regions to receive a transplant, perhaps because of financial limitations. Our "fractional-$\mathbf{d}$" result suggests that the system as a whole benefits even if only a small proportion of patients multiple list. This gives rise to questions about fairness: is the response time benefit experienced disproportionately by the redundant patients, or can patients who do not multiple list also benefit from others multiple listing? We leave such questions open for future work.

The observation that a little redundancy goes a long way is again important in computer systems, particularly when there may be some cost to creating multiple copies of jobs. For example, sending the same request to multiple servers might add network overhead, or cancelling the extra copies once the first copy completes might take some amount of time. While we do not explicitly model these costs, knowing that the most significant benefit comes from adding only one extra copy per job means that we can reduce response time without incurring too many of the corresponding costs. In many computer systems applications, the tail of response time is actually a much more critical metric than the mean. Our work provides the first analytical results showing how the tail of response time is influenced by redundancy in large systems.

Although Redundancy-$\mathbf{d}$ appears somewhat similar to dispatching policies such as JSQ-$\mathbf{d}$, our analysis of Redundancy-$\mathbf{d}$ is different. Dispatching policies such as JSQ-$\mathbf{d}$ and similar policies have only been studied in the limit as the number of servers approaches infinity using differential equations that typically track the fraction of queues with at least i jobs [13, 17, 19]. This is very different from our analysis, in which we track the amount of work in a queue as seen by a tagged arrival. Simply tracking the number of jobs in a queue is not powerful enough for analyzing the

Redundancy-$\mathbf{d}$ system because the departure process is much more complicated: it includes not only departures due to service completions at that server, but also departures due to completions of jobs' copies at *other* servers. We are hopeful that our approach in which we track the remaining work in a queue will open the door to analyzing more complicated "power of $\mathbf{d}$" dispatching policies.

One important assumption in our asymptotic analysis is that the queues are asymptotically independent. That is, knowing a job's "non-redundant" sojourn time at one server does not provide any information about what that same job's "non-redundant" sojourn time would be at a different server. This type of asymptotic independence is a very common precondition for the analysis of many related queueing systems. Unfortunately, the techniques typically used to prove asymptotic independence under related policies do not easily generalize to the Redundancy-$\mathbf{d}$ system, again because the departure process in the Redundancy-$\mathbf{d}$ system is very complicated. We leave the asymptotic independence assumption as a strongly supported conjecture; proving it remains open for future work.

Our analysis represents a first step towards solving several related queueing problems. For example, in an (n, k) system redundant copies of jobs are sent to multiple queues chosen at random, but more than one copy needs to complete [9, 10]. It is appealing to consider whether the analysis presented in this paper, which applies to the case where only a single copy needs to complete, can be extended to the general (n, k) system. Redundancy is also very closely related to fork-join systems, in which jobs send copies to all k servers and need all k copies to complete, and to coupled processor systems, in which multiple servers can work on the same job simultaneously. The fork-join and coupled processor problems are classically hard queueing problems; we hope that the analysis presented in this paper will inspire new approaches to these problems as well.

References

[1] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. Effective straggler mitigation: Attack of the clones. In *NSDI*, pages 185–198, April 2013.

[2] Ganesh Ananthanarayanan, Srikanth Kandula, Albert G Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris. Reining in the outliers in map-reduce clusters using mantri. In *OSDI*, volume 10, page 24, 2010.

[3] Mohammad Sanaei Ardekani and Janis M Orlowski. Multiple listing in kidney transplantation. *American Journal of Kidney Diseases*, 55(4):717–725, 2010.

[4] Barıs Ata, Anton Skaro, and Sridhar Tayur. Organjet: Overcoming geographical disparities in access to deceased donor kidneys in the united states. Technical report, Working paper, Northwestern University, 2012.

[5] Maury Bramson, Yi Lu, and Balaji Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Systems*, 71(3):247–292, 2012.

[6] Jeffrey Dean and Luis Andre Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, February 2013.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[8] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, Esa Hyytiä, and Alan Scheller-Wolf. Reducing latency via redundant requests: Exact analysis. In *SIGMETRICS*, June 2015.

[9] Gauri Joshi, Yanpei Liu, and Emina Soljanin. Coding for fast content download. In *Allerton Conference'12*, pages 326–333, 2012.

[10] Gauri Joshi, Yanpei Liu, and Emina Soljanin. On the delay-storage trade-off in content download from coded distributed storage systems. *IEEE Journal on Selected Areas in Communications*, 32(5):989–997, May 2014.

[11] Ger Koole and Rhonda Righter. Resource allocation in grid computing. *Journal of Scheduling*, 11:163–173, 2009.

[12] Robert M Merion, Mary K Guidinger, John M Newmann, Mary D Ellison, Friedrich K Port, and Robert A Wolfe. Prevalence and outcomes of multiple-listing for cadaveric kidney and liver transplantation. *American Journal of Transplantation*, 4(1):94–100, 2004.

[13] Michael Mitzenmacher. The power of two choices in randomized load balancing. *Parallel and Distributed Systems, IEEE Transactions on*, 12(10):1094–1104, 2001.

[14] Nihar B. Shah, Kangwook Lee, and Kannan Ramchandran. The MDS queue: Analysing latency performance of codes and redundant requests. Technical Report arXiv:1211.5405, November 2012.

[15] Nihar B. Shah, Kangwook Lee, and Kannan Ramchandran. When do redundant requests reduce latency? Technical Report arXiv:1311.2851, June 2013.

[16] Ashish Vulimiri, P. Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. Low latency via redundancy. In *CoNEXT*, pages 283–294, December 2013.

[17] N.D. Vvedenskaya, R.L. Dobrushin, and F.I. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Probl. Peredachi Inf.*, 32(1):20–34, 1996.

[18] Yunjing Xu, Michael Bailey, Brian Noble, and Farnam Jahanian. Small is better: Avoiding latency traps in virtualized data centers. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 7. ACM, 2013.

[19] Lei Ying, R Srikant, and Xiaohan Kang. The power of slightly more than one sample in randomized load balancing. In *Proc. of IEEE INFOCOM*, 2015.

[20] Matei Zaharia, Andy Konwinski, Anthony D Joseph, Randy H Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI*, volume 8, page 7, 2008.