How many servers are best in a dual-priority M/PH/k system?[1]

Adam Wierman Takayuki Osogami Mor Harchol-Balter Alan Scheller-Wolf
Computer Science Department and Tepper School of Business
Carnegie Mellon University

Abstract

We ask the question, "for minimizing mean response time (sojourn time), which is preferable: one fast server of speed 1, or k slow servers each of speed $1/k$?" Our setting is the M/PH/k system with two priority classes of customers, high priority and low priority, where PH is a phase-type distribution. We find that multiple slow servers are often preferable, and we demonstrate exactly how many servers are preferable as a function of the load and service time distribution. In addition, we find that the optimal number of servers with respect to the high priority jobs may be very different from that preferred by low priority jobs, and we characterize these preferences. We also study the optimal number of servers with respect to overall mean response time, averaged over high and low priority jobs. Lastly, we ascertain the effect of the service demand variability of high priority jobs on low priority jobs.

Keywords: Scheduling, queueing, multiserver, priority, preemptive, M/PH/k, dimensionality reduction, busy period.

1 Introduction

Is it preferable to use a single fast server of speed s, or k slow servers each of speed s/k? What is the optimal k?

The fundamental question of "how many servers are best" has been asked by a stream of research [22, 32, 18, 28, 21, 29] discussed in Section 2.2. All of this work considers the system where jobs are serviced in First-Come-First-Serve (FCFS) order, and asks both whether a single fast server is preferable to k slow servers (of equal total capacity) and how the optimal number of servers varies across workloads. We address these questions in the context of Poisson arrivals and phase-type (PH) service times and find that when load and service demand variability are high, multiple slow servers are preferable. It is easy to explicitly compute the optimal number of servers as a function of the load and service demand variability. We show that using more slow servers can sometimes reduce mean response time (sojourn time) by an order of magnitude, whereas using too many slow servers can have the opposite effect.

The above result is not all-encompassing, however, since real-world systems are often not simply FCFS, where all jobs have equal importance. Rather, there is often inherent scheduling in the system to allow high priority jobs to move ahead of low priority jobs. For example, the high priority jobs may carry more importance, e.g., representing users who pay more. Alternatively, high priority jobs may simply consist of

[1]This work was supported by NSF Grant CCR-0311383, and grant sponsorship from IBM Corporation.

those jobs with small service demand, since it is well-known that giving priority to short jobs reduces mean response time overall. Either way, priorities are fundamental to many modern systems.

This motivates the question of what the optimal system configuration is (a few fast servers or many slow servers) in a setting where there are different priority classes. In this paper we specifically assume two priority classes, a PH service demand distribution for each class, Poisson arrivals, and FCFS service order within each class.

The difficulty in answering the above question under the dual-priority setting is that the analysis of mean response time for the M/PH/k with two priority classes is not known, despite the tremendous amount of attention in the literature given to this problem in the context of exponentially distributed service demands. In the first half of our paper, we present a new approach for analyzing the M/PH/k with dual-priority classes. Our approach extends a technique used, for example, in [10], which we call "dimensionality reduction." This technique allows one to track two job classes using only a 1-dimensionally infinite Markov chain rather than the standard 2-dimensionally infinite Markov chain. Unfortunately, the existing dimensionality reduction technique does not apply to the problem of the M/PH/k dual-priority queue because of complications arising from the PH service times (see Section 4.1). A major part of our extension of dimensionality reduction involves invoking a method of Neuts [23] for finding moments of passage times in semi-Markov processes. We will use this method to derive specialized busy period durations, opening up a whole new class of problems that dimensionality reduction can now solve. Although our analysis is an approximation, it can be made as accurate as desired, and throughout the paper the accuracy is within a few percent of simulation.

Armed with a near-exact analysis of the M/PH/k dual-priority queue, for the remainder of the paper we focus directly on questions involving choosing the optimal resource configuration. In particular we are interested in the following questions:

1. Under what conditions are multiple slow servers preferable to a single fast server? Is the optimal number of servers sensitive to changes in the relative arrival rates of the priority classes, changes in the variability of the service time distributions, etc.?

2. Does the answer to "how many servers are optimal" differ for the different priority classes? E.g., does the lower priority class prefer more or fewer servers than that preferred by the higher priority class?

3. How does the optimal number of servers in a *dual* priority system differ from the case when all jobs have been aggregated into a *single* priority class?

4. If one chooses a non-optimal number of servers, how does that affect the overall mean response time and the per-class mean response time?

In Section 2, we describe prior methods for analyzing the M/M/k dual-priority queue. We also discuss prior work dealing with our question of "how many servers are best," most of which assumes exponential service demands, and *none* of which deals with multiple priority classes. In Section 3, we answer question 1 above in the case of an M/PH/k/FCFS queue with a single priority class. Section 4 presents our new analysis of the M/PH/k queue with dual-priority classes and the validation of our analysis against simulation. In Section 5, we address all four questions above for the case of an M/PH/k dual-priority queue. Finally, we conclude in Section 6.

2 Prior work

Section 2.1 will discuss prior work on multiserver systems with two priority classes. While the literature is vast, almost all is limited to exponential service times. Section 2.2 discusses prior work dealing with questions related to the optimal number of servers. Here too, the literature is vast, although *all* deals with only a single priority class, and the majority focuses on exponential service times.

2.1 Dual priority analysis in multiserver systems

2.1.1 Exponential service times

Almost all the papers written on dual-priority multiserver systems are restricted to exponential service times (M/M/k). Techniques for analyzing the M/M/k dual-priority system can be organized into four types on which we elaborate below: (i) approximations via aggregation or truncation; (ii) matrix analytic methods;

(iii) generating function methods; (iv) special cases where the priority classes have the same mean. Unless otherwise mentioned, preemptive-resume priorities should be assumed.

Nearly all analysis of dual-priority M/M/k systems involves the use of Markov chains, which with two priority classes grows infinitely in two dimensions (one dimension for each priority class). In order to overcome this, researchers have simplified the chain in various ways. Kao and Narayanan truncate the chain by either limiting the number of high priority jobs [12], or the number of low priority jobs [13]. Nishida aggregates states, yielding an often rough approximation [25]. Kapadia, Kazmi and Mitchell explicitly model a finite queue system [15]. Unfortunately, aggregation or truncation is unable, in general, to capture the system performance as the traffic intensity grows large.

Although, in theory, the matrix analytic method can be used to directly analyze a 2D-infinite Markov chain (see for example [2]), the matrix analytic method is much simpler and more computationally efficient when it is applied to a 1D-infinite Markov chain. Therefore, most papers that use the matrix analytic method to analyze systems involving 2D-infinite Markov chains first reduce the 2D-infinite chain to a 1D-infinite chain by, for example, truncating the state space by placing an upper bound on the number of jobs [12, 13, 17, 24]. Miller [19] and Ngo and Lee [24] partition the state space into blocks and then "super-blocks," according to the number of high priority customers in queue. This partitioning is quite complex and is unlikely to be generalizable to non-exponential job sizes. In addition, [19] experiences numerical instability issues when $\rho > 0.8$.

A third stream of research capitalizes on the exponential job sizes by explicitly writing out the balance equations and then finding roots via generating functions. In general these yield complicated mathematical expressions susceptible to numerical instabilities at higher loads. See King and Mitrani [20]; Gail, Hantler, and Taylor [8, 9]; Feng, Kowada, and Adachi [7]; and Kao and Wilson [14].

Finally there are papers that consider the special case where the multiple priority classes all having the same mean. These include Davis [6], and Buzen and Bondi [3].

2.1.2 Non-exponential service times

The few papers which deal with non-exponential service time are either coarse approximations or only handle limited settings.

In a follow-up to [3], Bondi and Buzen extend their technique to general service distributions and multiple classes [1]. As no exact results exist, they compare their approximations against simulation. In order to understand how our results compare with those in [1], we will plot our results against their approximations directly in Section 4.3, showing that our techniques yield much more accurate results.

The only work dealing with non-exponential service times in a more precise setting is contained in a pair of papers, not yet published, by Sleptchenko [30] and Sleptchenko et. al. [31]. Both papers consider a two-priority, multiserver system where within each priority there may be a number of different classes, each with its own different exponential job size distribution. This is equivalent to assuming a hyperexponential job size distribution for each of the two priority classes. The problem is solved via a combination of generating functions and the matrix analytic method. In theory, their technique can be generalized to PH distributions, though they evaluate only hyperexponential distributions due to the increased complexity necessary when using more general PH distributions.

2.2 How many servers are best in an M/GI/k?

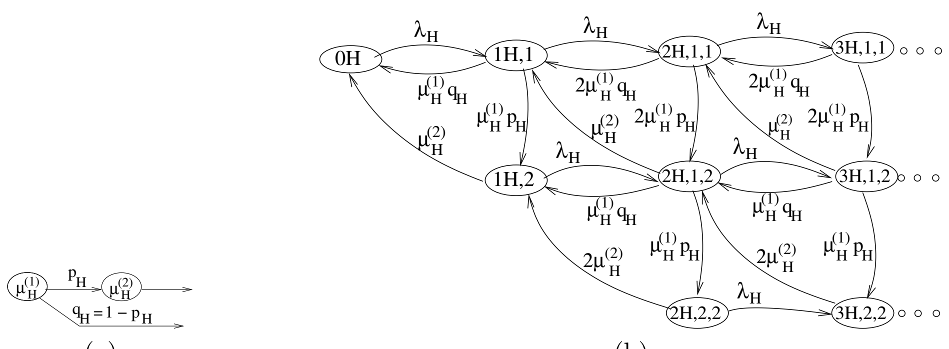
The question of how many servers are best has a long history, all assuming a single priority class. In describing the results below, and throughout the paper, we will assume that the performance metric of interest is mean response time rather than mean queueing time, since it's obvious that to minimize mean queueing time one wants an infinite number of servers [4, 5].

As early as 1958 Morse observed that for an M/M/k system the optimal number of servers is one [22]. This was formalized by Stidham [32], who showed that under a general arrival process, and service times that are exponential, Erlang, or deterministic, a single server minimizes the expected number in system. Likewise, for a single server, Mandelbaum and Reiman [18] prove that one server is best in the extreme cases when traffic is very light, regardless of job size variability. So, not only is variability important, but so too is traffic intensity. Scheller-Wolf [28] characterizes the effect of traffic intensity. He shows that under so-called power-law service times, moments of response time may move from infinite to finite as a function of both the number of servers and the traffic intensity. Very recently, Molinero-Fernandez et. al. [21] consider the question of how many servers are best in an M/HT/k single priority system, where HT denotes a *heavy-tailed* service distribution. To answer this question, they approximate a heavy-tailed distribution with a bimodal distribution (BM), and then provide an approximate closed-form analysis of the M/BM/k queue, which they argue provides a reasonable approximation of the M/HT/k queue. The question of how many servers is best has also been considered via simulation in the context of an M/G/k queue by [29]. None of the above work considers priorities.

This paper is the first to characterize the optimal number of servers for two priority classes under PH job size distributions. Before presenting the results, we begin with systems having only a single priority class, which will be utilized in our analysis of the dual-priority system.

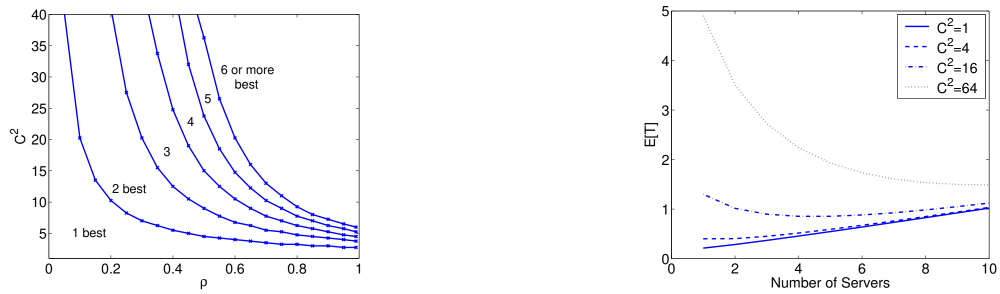
3 Single priority class: How many servers are best?

In this section, we consider the simplified problem of determining the number of servers that minimize the mean response time under just *one* priority class. Although no exact analysis exists for the M/GI/k/FCFS



(a) (b)

Figure 1: *Figure illustrates that the M/PH/k/FCFS queue (single class – i.e., all jobs are high priority) is easy to analyze. (a) shows a 2-phase PH distribution with Coxian representation. (b) shows the Markov chain that is used to compute the mean response time for an M/PH/2 queue, where PH has the representation shown in (a). The state (nH, i, j) represents the number of jobs in the system (nH) and the current phase of each job in service (i and j). This the generalization of this chain to an arbitrary PH service distribution is straightforward.*



(a) (b)

Figure 2: *The case of a single priority class. (a) The optimal number of servers as a function of the load, ρ, and the variability of the job size distribution, C^2. (b) Mean response time, $E[T]$, as a function of the number of servers at various job size variabilities ($C^2 = 1, 4, 16, 64$) for a fixed $\rho = 0.6$.*

queue, the M/PH/k/FCFS queue is easily analyzable via matrix analytic methods [16], as its Markov chain has a state space infinite in only one dimension. Figure 1 shows a picture of the Markov chain that we use for analyzing an M/PH/2/FCFS queue using the matrix analytic method.

Figure 2(a) shows the optimal number of servers as a function of the load and the variability of the job size. All of our results are expressed as a function of the variability of the job size distribution, and the server load. While other factors, e.g., the exact form of the distribution might affect our results, we posit that load and variability will be the most relevant factors.

Observe that under high job size variability and/or high load, the optimal number of servers is more than 1; we prefer k slow servers to 1 fast server. For example, at load $\rho = 0.4$ and $C^2 = E[X^2]/E[X]^2 - 1 = 20$, we see that 3 servers are best. Computations are only done for up to 6 servers — the level curves shown will continue into the upper right portion of the plot if larger systems are considered.

Figure 2(b) shows that for any particular job size variability, $C^2 > 1$, having a larger number of slower servers may reduce the mean response time up to a point, after which further increasing the number of servers increases the mean response time. To understand why, note that by increasing the number of servers (while maintaining fixed total capacity), we are allowing short jobs to avoid queueing behind long jobs — specifically, an arriving short job is more likely to find a server free. Thus increasing the number of servers mitigates variability, hence improving performance. If the number of servers is too great however, servers are more likely to be idle, under-utilizing the system resources.

This simple analysis of the single priority M/PH/k queue motivates questions about how having two priority classes changes the answer to the question of "how many servers?" and whether approximating the more complicated two priority system with a single priority system is feasible. These questions are central to the remainder of this work.

4 Analysis of the M/PH/k with dual priorities

In this section, we describe our analysis of the mean response time in M/PH/k/FCFS queues having two priority classes (high priority and low priority), where high priority jobs have preemptive-resume priority over low priority jobs. Since the mean response time of the high priority jobs can be analyzed as an M/PH/k/FCFS queue with a single priority class (as in the previous section), we concentrate here on the mean response time of the low priority jobs. Our goal in this section is to present a complete, but intuitive, description of the analytic technique used. For a formal derivation of the analysis see [11] or [26], where the analytic technique is also extended to apply for $m > 2$ priority classes and more general multiserver systems.

As mentioned in Section 2, the Markov chain for our system grows infinitely in two dimensions, which makes analysis via standard techniques problematic. The PH service times further complicate the model, invalidating existing dimensionality reduction techniques such as that in [10]. We explain this below in Section 4.1.

Following the description of the analytic technique, we validate our analysis using simulations in Section 4.3. Further, while performing validations, we evaluate the performance of known approximations for the M/GI/k dual-priority queue.

4.1 Analysis of the M/M/k with dual priorities

We will illustrate standard dimensionality reduction in the case of two servers and two priority classes, where both high and low priority jobs have exponentially-distributed sizes with rate μ_H and μ_L respectively. We will then describe how standard dimensionality reduction fails in the more general case of PH job sizes.

Figure 3 (a) illustrates a Markov chain of the M/M/2 dual-priority system. The states of the chain track the number of high priority and low priority jobs, and hence the chain grows infinitely in two dimensions. Observe that high priority jobs can be analyzed independently of low priority jobs, but low priority jobs need to be analyzed together with high priority jobs since their behavior depends on the number of high priority jobs.

Figure 3 (b) and (c) illustrate the reduction of the 2D-infinite Markov chain to a 1D-infinite Markov chain via dimensionality reduction. The 1D-infinite chain tracks the number of low priority jobs exactly. For the high priority jobs, the 1D-infinite chain only differentiates between zero, one, and two-or-more high

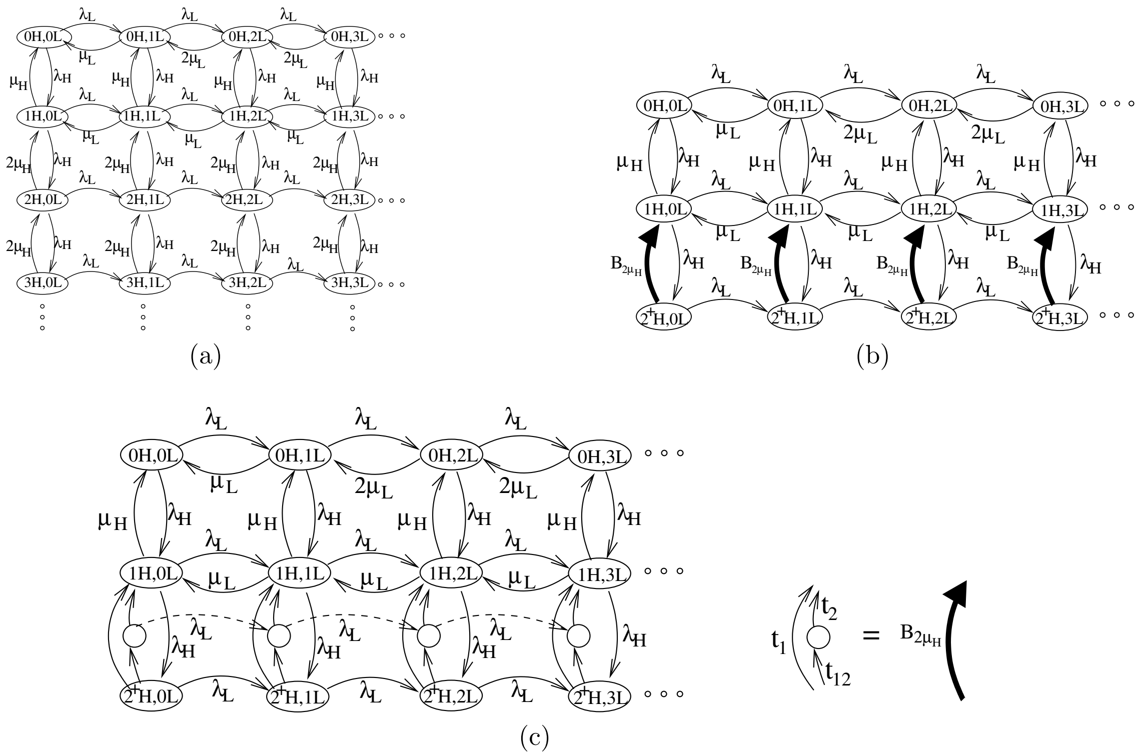


Figure 3: *Markov chain for a system with 2 servers and 2 priority classes where all jobs have exponential sizes. The chain in (a) is infinite in two dimensions. The chain in (b) uses busy period transitions and is only infinite in one dimension. In (b), the busy period is represented by a single transition. In (c), the busy period is represented by a two phase PH distribution (with Coxian representation).*

priority jobs since this information is necessary and sufficient to capture the behavior of low priority jobs. The distribution of the sojourn time in each row (0, 1, or 2+ high priority jobs) also needs to be calculated. In particular, the length of time spent with at least two high priority jobs in the system is exactly a single-server M/M/1 busy period where the service rate is $2\mu_H$. In Figure 3 (b), we denote the length of this busy period by the transition labelled $B_{2\mu_H}$. We use a PH distribution to match the first three moments of the distribution of $B_{2\mu_H}$[2], as in Figure 3 (c). The limiting probabilities of the Markov chain in Figure 3 (c) can be analyzed using the matrix analytic method. Given the limiting probabilities, it is easy to obtain the mean number of low priority jobs, which in turn yields their mean response time from Little's law.

Figure 4 shows the generalization to a 3-server system. We have added one row to the Markov chain shown in Figure 3 (b), and we now differentiate between 0, 1, 2, or 3-or-more high priority jobs. As in Figure 3, the transitions labelled $B_{3\mu_H}$ can be replaced by PH distributions. The analysis can be readily extended to the case of $k > 3$ servers.

This analysis of M/M/k dual-priority systems relies solely on standard dimensionality reduction techniques; however, this analysis technique does not easily generalize to PH high priority service times. Although

[2]Matching three moments of busy period distributions is often sufficient to guarantee accurate modeling of many queueing systems with respect to mean performance [27].

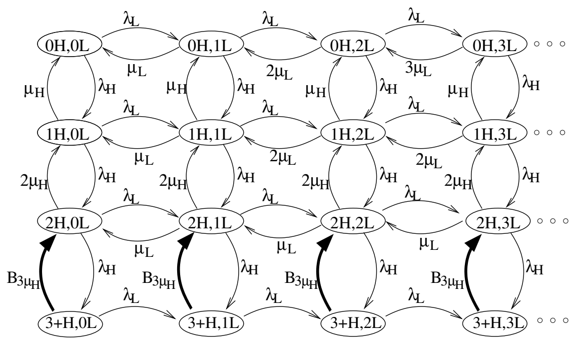


Figure 4: *A 1D infinite Markov chain for the case of 2 priority classes and 3 servers where job sizes have exponential distributions.*

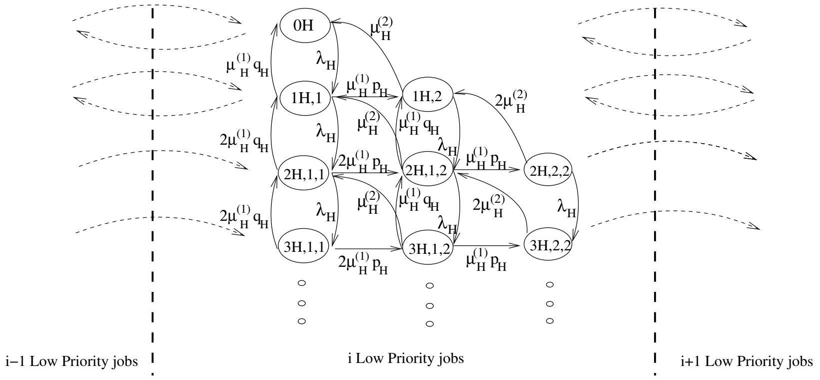


Figure 5: *One level of the two dimensionally infinite Markov chain for a system with 2 servers and 2 priority classes where high priority jobs have PH service times with Coxian representation shown in 1(a).*

it may at first appear that such a generalization is possible, the difficulty lies in the fact that we can no longer view the busy period, which starts when the number of high priority jobs goes from $k-1$ to k and ends when the it drops back to $k-1$, as simply an M/M/1 busy period with service rate $k\mu_H$.

There are two complications that make extending the method to PH job sizes difficult. These complications are illustrated in Figure 5, where the two dimensionally infinite chain is shown. First, the "busy period" that we need in the system with PH job sizes is much more complex than an M/M/1 busy period. The length of a busy period in a system with PH job sizes depends on the phases of the jobs being served at both the beginning and the end of the busy period. The busy period thus can no longer be represented by a single distribution – a new representation is needed. Second, in order to parameterize a representation of this new type of busy period, we need to be able to analyze its length. Prior dimensionality reduction has relied on closed-form analyses for the duration of M/GI/1 busy periods, which is not possible in this case.

4.2 Extension to the M/PH/k with dual priorities

As we just observed, although it appears at first that the Markov chains shown in Figures 3 and 4 might easily generalize to the case of PH service times, the generalization is not trivial. In the case where the

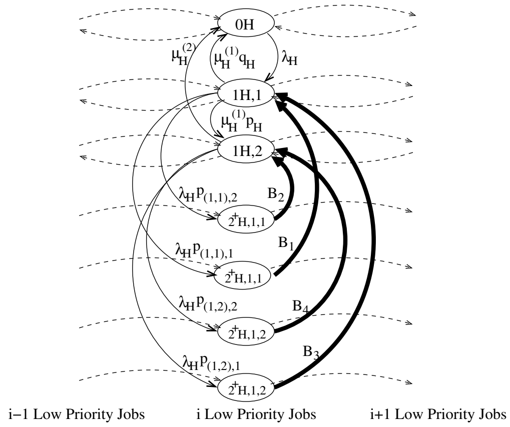
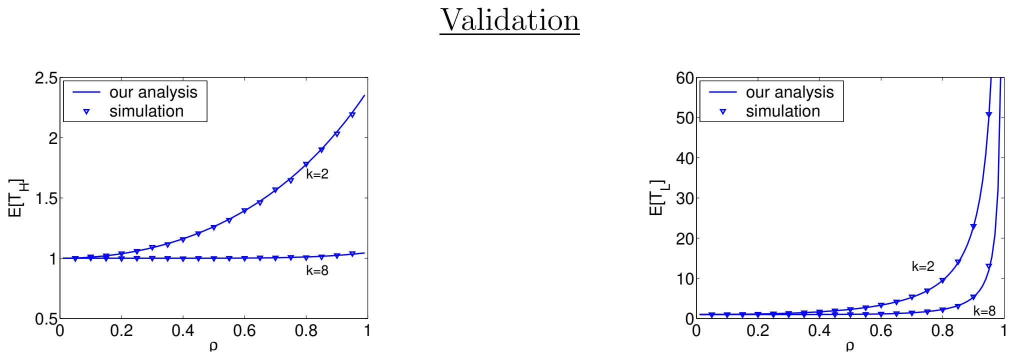


Figure 6: *One level of a one dimensionally infinite Markov chain using generalized busy period transitions to represent a system with 2 servers and 2 priority classes where high priority jobs have PH service times with Coxian representation shown in 1(a). This is a reduction of the two dimensionally infinite chain shown in Figure 5 to a one dimensionally infinite chain.*

number of servers is two ($k = 2$) and the high priority jobs are represented by 2-phase PH distribution with Coxian representation, there are four different types of busy periods, depending on the phases of the two jobs starting the busy period and the phase of the job left at the end of the busy period (Figure 5). These busy periods are either (i) started with two jobs both in phase 1 and ended with a job in phase 1, (ii) started with two jobs both in phase 1 and ended with a job in phase 2, (iii) started with one job in phase 1 and another in phase 2 and ended with a job in phase 1, or (iv) started with one job in phase 1 and another in phase 2 and ended with a job in phase 2. (Note that a busy period can never start with two jobs both in phase 2.)

Once the distributions of the four types of busy periods are approximated by PH distributions the 2D-infinite Markov chain can be reduced to a 1D-infinite Markov chain. The 1D-infinite Markov chain for PH job sizes tracks the exact number of low priority jobs but only differentiates between 0, 1, and 2-or-more high priority jobs for the case of two servers ($k = 2$). Figure 6 shows the level of the Markov chain, where the number of low priority jobs is i. In state (0H), no high priority jobs are in system; in state (1H,u), one high priority job in phase u is in system for $u = 1, 2$; in state (2^+H,u, v), at least two high priority jobs are in system (we are in a high priority busy period), and the two jobs that started the busy period were in phases u and v, respectively, for $u = 1, 2$ and $v = 1, 2$. The four types of busy periods are labelled as B_1, B_2, B_3, and B_4; the durations of these busy periods are approximated by PH distributions matched to the first three moments of the distribution. Finally, $p_{(u,v),w}$ denotes the probability that a busy period started by two jobs in phases u and v respectively ends with a single job in phase w for $u = 1, 2$, $v = 1, 2$, and $w = 1, 2$. Note that the ending phase of the busy period can be determined probabilistically at that moment when the second high priority job arrives which starts the busy period. The length distribution is then modeled conditionally on this ending phase.

The remaining question is how to analyze the probabilities $p_{(u,v),w}$ and the first three moments of the duration of busy periods, B_1, B_2, B_3, and B_4. Observe that the Markov chain for high priority jobs (Figure



(a) High Priority Response Time

(b) Low Priority Response Time

Figure 7: *Validation of our analysis against simulation: two class M/PH/k system where* $\rho_L = \rho_H$ *and* $E[X_L] = E[X_H] = 1$. *The high priority jobs have a 2-phase PH job size distribution with* $C^2 = 9$, *shown in Figure 1(a), and the low priority jobs have exponential service times.*

1(b)) is a QBD process. Moreover, due to the way we have modeled the system, the durations of different types of busy periods correspond to the first passage time to state (1H,w) from state (2H,u,v) for different values of u, v, and w. Likewise, probability $p_{(u,v),w}$ corresponds to the probability that state (1H,w) is the first state reached in level 1 given that we start from state (2H,u,v) for $u = 1, 2$, $v = 1, 2$, and $w = 1, 2$. Neuts' technique [23] establishes just these quantities for general QBDs and semi-Markov processes.

Our goal in this section is only to present an overview of the analytic technique used. For a formal derivation of the technique see [11] or [26], where the analytic technique is extended to apply for $m > 2$ priority classes and the details of Neuts' technique for deriving passage times in QBD processes are explained.

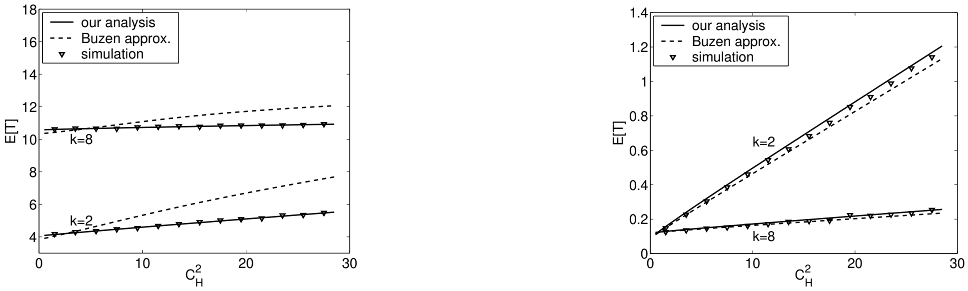
4.3 Validation and comparison with prior work

The purpose of this section is twofold: (i) we validate our analysis against simulation, and (ii) we compare the results of our analysis to known approximations for the M/GI/k dual-priority queue. In addressing the above, we also begin our investigation of the factors that affect the mean response time of high and low priority jobs.

In order to validate our algorithm, we compare our calculations with simulated results under a wide variety of loads and service distributions. Our simulations were calculated by running over 100,000 events for each of 30 iterations and averaging the results.

A subset of these validations are shown in Figure 7, for the case of 2 servers and 8 servers. Throughout this section low priority jobs have exponential job sizes and high priority jobs have PH job sizes with $C^2 = 9$. As shown, our analysis closely matches simulation (typically $< 1\%$, always within a few percent) for both the high priority and low priority jobs. Because our analysis uses the matrix analytic method, the computational cost of the algorithm is well understood in terms of the size of the repeating matrices, which here is $k(5k+1)/2$ for a k server system, where high priority job sizes have a 2-phase PH distribution with Coxian representation. In practice this corresponds to a computation time of less than 0.1 seconds to calculate the mean response

Comparison: Mean response time



(a) $E[X_H] = 1$, $E[X_L] = 10$

(b) $E[X_H] = 1$, $E[X_L] = \frac{1}{10}$

Figure 8: *Comparison of our analysis with simulation and the Buzen-Bondi approximation for M/GI/k with dual-priority. Only response times for low priority jobs are shown. (a) shows case when high priority jobs have a smaller mean job size than low priority jobs: for $k = 8$, $E[X_H] = 1$, $E[X_L] = 10$ and for $k = 2$ the job sizes are scaled appropriately (by a factor of 4). (b) shows case when high priority jobs have a larger mean job size than low priority jobs: for $k = 8$, $E[X_H] = 1$, $E[X_L] = \frac{1}{10}$. In these plots $\rho = 0.6$.*

time of a system where $E[X_H] = E[X_L] = 1$ and $\rho_H = \rho_L = 0.25$ with $k = 4$.

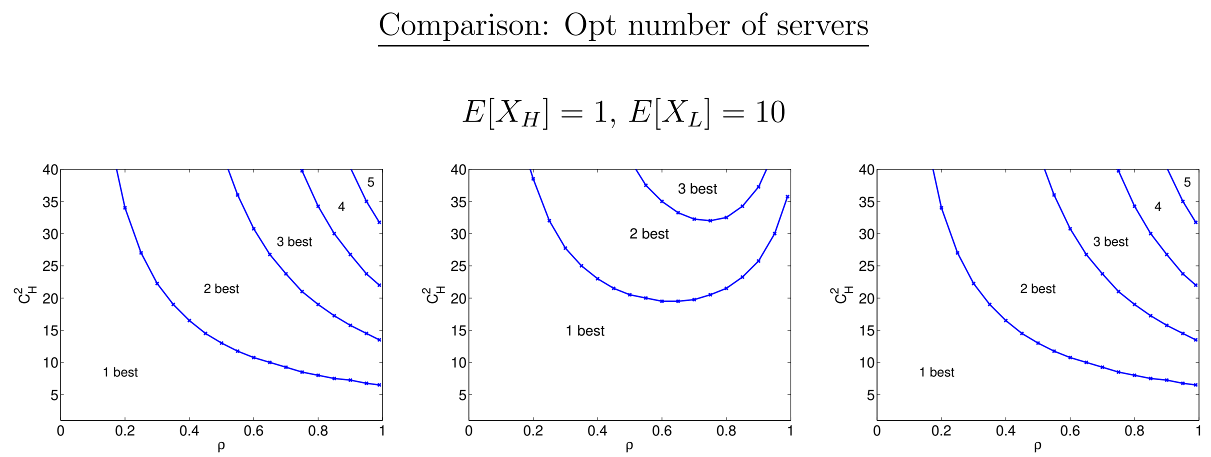
It is interesting to compare the results of our analysis with the Buzen-Bondi approximation [1], which is the only published prior work to investigate the 2 class system under general job size distributions. Buzen and Bondi use the following intuitive framework to approximate the performance of a priority system, where $E[W]^{M/GI/k/Prio}$ is the overall expected waiting time (response time minus service time) under the M/GI/k queue with priority classes.

$$E[W]^{M/G/k/Prio} = E[W]^{M/G/1/Prio} \left(\frac{E[W]^{M/G/k/FCFS}}{E[W]^{M/G/1/FCFS}} \right)$$

In our comparison, we use the exact results for $E[W]^{M/G/k/FCFS}$ (see Section 3); thus, the mean response time for the high priority jobs will be exact. However, when looking at low priority jobs, we still see significant improvement of our algorithm over the Buzen-Bondi approximation.

Figure 8 shows the mean response time for low priority jobs predicted by our analysis, the Buzen-Bondi approximation, and simulation. Figure 8 (a) shows the case where high priority jobs have a smaller mean size than low priority jobs; and Figure 8 (b) shows the converse. Our analysis does significantly better than the Buzen-Bondi approximation, particularly when high priority jobs have a smaller mean size. As we increase the variability of the high priority job sizes, the error of the Buzen-Bondi approximation increases, while our analysis matches with simulation for all cases. Although not shown, the same trends also hold as the system load is varied.

The consequence of the mispredictions made by the Buzen-Bondi approximation may be important. The error in predicting the response time can lead to a misleading conclusion on the optimal number of servers (a question we will investigate in detail in Section 5). Figure 9 shows the optimal number of servers given by our analysis as compared with both simulation and that predicted by the Buzen-Bondi approximation. The



(a) Our analysis (b) Buzen-Bondi (c) Simulation

Figure 9: *Comparison of our results with that of Buzen-Bondi with respect to the question of "how many servers are best?" The case shown assumes the high priority jobs have a smaller mean job size. The load made up by high priority jobs equals that comprised by low priority jobs. Column (a) shows the results from our analysis; column (b) shows the results from the Buzen-Bondi approximation; and column (c) shows the results from simulation.*

Buzen-Bondi approximation is not only incorrect at higher loads and/or C^2 values, but also qualitatively misleading. As load increases, so should the optimal number of servers; an effect not true under the Buzen-Bondi approximation (Figure 9(b)). We explore this in the next section.

In addition to validating our analysis, Figures 7 and 8 also illustrate some important factors affecting mean response time of high and low priority jobs: the system load, the variability of high priority customers, and the number of servers. Observe for example that for the job size distributions in Figure 8 (a) 2 servers are preferable, while in (b) 8 servers are preferable. The effects of these factors are interrelated and will be investigated in detail next.

5 How many servers are best?

In the introduction to this paper we set out to answer four questions. We have already addressed the first of these. In answer to question 1, we have seen that in both the case of single priority class (Section 3) and in the case of dual-priority classes (Section 4.3) multiple slow servers can be preferable to a single fast server. Further, we have seen that the preference depends on service time variability and system load. For the case of a single priority class, this dependence is shown in Figure 2 and for the case of dual-priority classes this is shown in Figure 9. The reason why multiple slow servers are preferable under high variability job sizes is that they offer short jobs a chance to avoid queueing behind long jobs, which in turn lowers mean response time.

We will now focus our investigation on the three remaining questions from the introduction. (Question 2) How does the answer the the question of "how many servers are optimal" differ among priority classes?

(Question 3) How does a dual-priority system differ from its corresponding aggregate single priority system in terms of the optimal number of servers? (Question 4) How much improvement in mean response time can choosing the optimal number of servers provide?

We find that the answers to these questions depend on the relative sizes and relative proportions (loads) of the classes, as well as the variability of high priority jobs. We briefly summarize some of our findings. The number of servers preferred by low priority versus high priority jobs can vary widely; we perform a sensitivity analysis to characterize exactly when they disagree. Moreover, the number of servers preferred in the dual-priority case when averaged over both classes typically differs substantially from the number preferred for the single class aggregate case. Furthermore, the absolute effect on mean response time can be dramatic (ranging from a factor of 2 to 6) as the number of servers is varied. In all studied cases, there exists an "optimal" number of servers where using fewer or more servers results in worse performance under highly-variable service distributions.

5.1 Setup for results graphs

We split up our evaluation into 3 cases, depending on the relative sizes of high and low priority jobs:

(i) The mean size of high priority jobs equals that of low priority jobs: $E[X_H] = 1$, $E[X_L] = 1$.

(ii) The mean size of high priority jobs is smaller than that of low priority jobs: $E[X_H] = 1$, $E[X_L] = 10$.

(iii) The mean size of high priority jobs is larger than that of low priority jobs: $E[X_H] = 1$, $E[X_L] = 1/10$.

Note that the mean service time changes depending on how many servers are in the system (1 fast server or k slow servers) so that the systems are comparable. The values specified are the values for the maximum number of servers used in each plot and the mean sizes for each of the other number of servers is scaled appropriately.

Throughout our evaluations, we will consider a range of variability in the high priority job sizes, typically shown on the y-axis, and a range of load typically shown on the x-axis. The variability of the low priority job sizes is held constant ($C^2 = 1$). Observe that variability in the low priority jobs is less interesting since the low priority jobs only affect each other under preemptive resume. Lastly we also vary the proportion of the load made up by high priority and low priority jobs.

Some technicalities of the setup follow. In all the results shown, the high priority job sizes follow a 2-phase PH distribution with Coxian representation (Figure 1(a)), allowing any variability $C^2 \geq 1.5$. The mean job size for high priority jobs is held fixed at 1. When varying the proportion of load in each priority class, we vary the arrival rates of the classes only. In order to compare our results for the dual-priority system to the same system having a single aggregate class, we use a mixture of the two-phase PH high priority job size distribution and the exponential low priority job size distribution to obtain the overall job size distribution aggregated across both classes.

5.2 Results

Figures 10, 11, and 12 illustrate the results of our analysis for the three cases above: (i) $E[X_H] = 1$, $E[X_L] = 1$; (ii) $E[X_H] = 1$, $E[X_L] = 10$; and (iii) $E[X_H] = 1$, $E[X_L] = 1/10$ respectively. For each figure column (a) shows the case where the load made up by high and low priority jobs is equal, and column (b) shows the case where $\rho_H < \rho_L$. We also discuss, but omit showing, the case where $\rho_H > \rho_L$. For each figure we consider both the case of dual-priority classes and the case of a single aggregate class.

Figure 10: Equal mean sizes

Looking at the topmost plot in Figure 10 column (a), we see that the high priority jobs do not always prefer one server. In fact in the case of higher variability and/or load, they may prefer five or more servers. This is to be expected based on our results in Section 3.

Surprisingly however, the number of servers preferred by low priority jobs (shown in the second plot in column (a)) is much greater than that preferred by high priority jobs. Although only up to six servers are considered in these plots, we will see in later plots (Figure 13(b)) that the difference in the number of servers preferred by low and high priority jobs can be more than 10 servers. Low priority jobs prefer more servers because low priority jobs are preempted by high priority jobs and thus their mean response time improves with more servers, which allows them to escape from the dominance of high priority jobs.

The preferred number of servers with respect to the overall mean response time (the average of all jobs, including both low and high priority jobs) is shown in the third plot in column (a), where we see that the number of servers preferred by the overall mean, as expected, is a hybrid of that preferred by low and high priority jobs. Note though that this hybrid is more weighted toward the preference of low priority jobs because adding extra servers only hurts high priority jobs a small amount; whereas adding extra servers helps low priority jobs enormously. Interestingly, the number of servers preferred with respect to the overall mean is nearly identical to that shown for a single aggregate class of high and low priority jobs, shown in the bottom most plot in column (a). To understand why, observe that all jobs in this case have the same mean, and thus prioritizing in favor of some of them over others does not affect the mean response time greatly. Even though the classes have different variabilities, that is a smaller-order effect. This will not remain true in general.

Moving to column (b) of the same figure, we see that the same trends are evident when the high priority jobs make up a smaller fraction of the load. However, the specific numbers are quite different. For example, starting with the topmost plot in column (b), we see that the number of servers preferred by high priority jobs is much fewer. An explanation of this is that the high priority jobs only interfere with each other and they are fewer in number in column (b) than in column (a); thus they want fewer, faster servers.

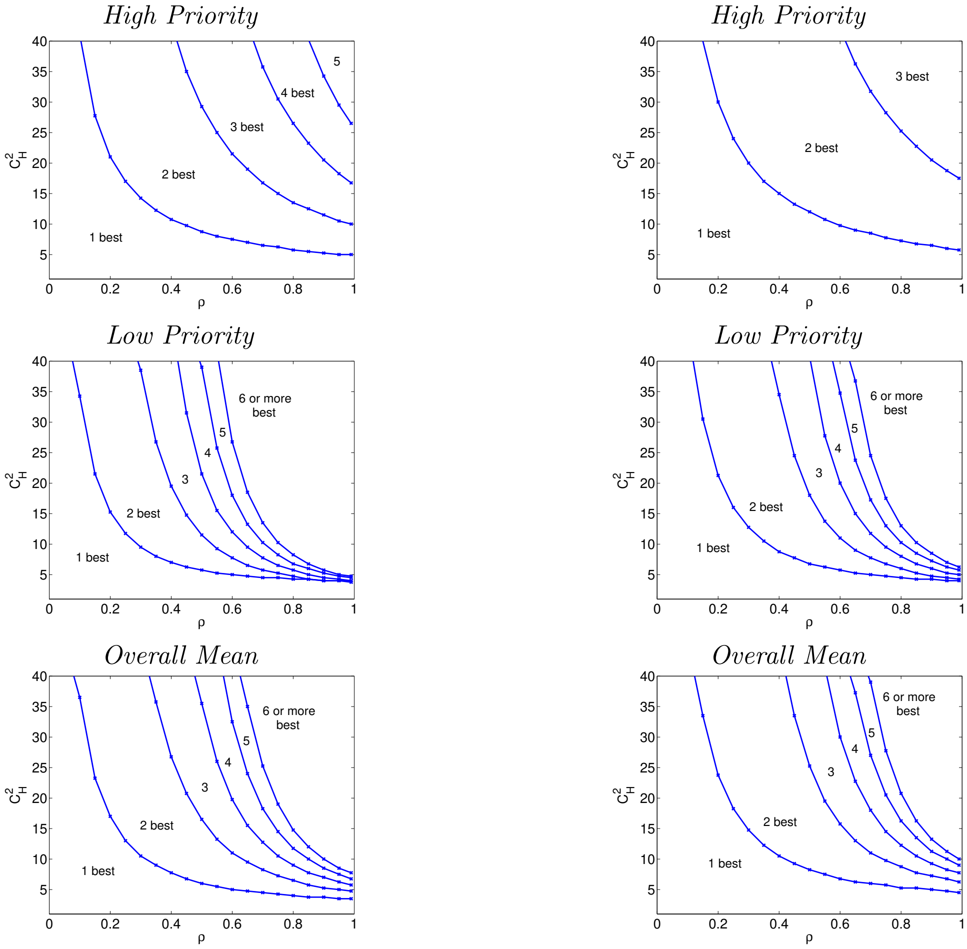
Less obvious is the fact that the number of servers preferred by low priority jobs in column (b) is also fewer than that in column (a). This follows from the same reasoning; the low priority jobs are most strongly affected by preemptions from high priority jobs, and with fewer high priority jobs, there are fewer interruptions and thus fewer servers are needed to avoid queueing behind high priority jobs.

Since both the high and low priority jobs in column (b) prefer fewer servers than in column (a), it makes sense that their overall mean (shown in the third plot of column (b)) also indicates that fewer servers are desired. This third plot also matches the bottom most plot in column (b) consisting of a single-class aggregation of high and low priority jobs, for the same reason explained above – that jobs have the same mean.

Not shown in Figure 10 is the case where high priority jobs comprise more of the load. In this case, both

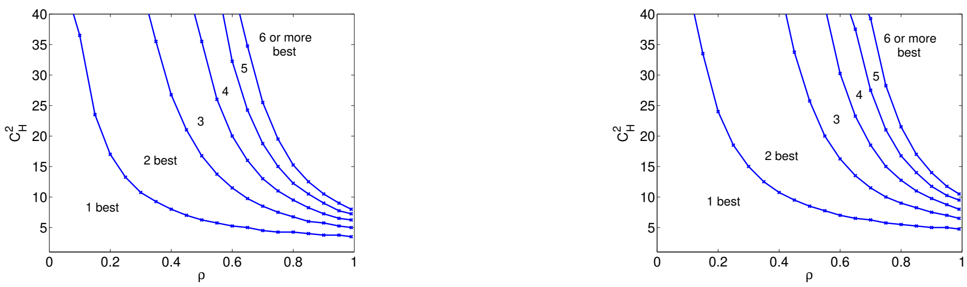
(i) $E[X_H] = 1,\ E[X_L] = 1$

2 Priority Classes



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

1 Aggregate Class



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

Figure 10: *How many servers are best (with respect to mean response time) when the two priority classes have the same mean job size? Column (a) shows the case where the load made up by high priority jobs equals that of low priority jobs. Column (b) shows the case where the load made up by high priority jobs is half that of low priority jobs.*

classes prefer more servers and, therefore, the mean of the two classes also prefers more servers. The reason for this is the converse of the above situation – there are more high priority jobs, and therefore they see more interference and want more servers. Further, the low priority jobs are preempted more frequently by high priority jobs and therefore also want more servers to alleviate the effect. Again the single aggregate class looks very similar to the two priority class overall mean.

Figure 11: High priority class has smaller mean

Moving to Figure 11, we continue to hold the mean high priority job size at 1 and increase the low priority job size to 10. Here, giving high priority jobs preference schedules the system more efficiently.

Notice that the preferred number of servers for the high priority jobs is identical to that in Figure 10 because the high priority job size distribution is unchanged. However, the number of servers preferred by low priority jobs is now very different: they almost always prefer only one server. This follows from the fact that there are very few low priority jobs; so there is unlikely to be more than one low priority job in the system at a time. Thus, low priority jobs prefer a single fast server.

The overall preferred number of servers, averaged over the two priority classes, is again a hybrid of the preferences of the two classes, but this time is biased toward the preferences of the high priority jobs because they are in the majority, implying a preference for fewer servers than the corresponding graph in Figure 10. Recall that adding servers is a way to help small jobs avoid queuing behind larger jobs. Since we are in the case where small jobs have priority already, we do not need the effect of multiple servers. *Thus, in this case, priority classes can be viewed as a substitute for adding more servers.*

Comparing the overall preferred number of servers for the case of dual priorities with that preferred under a single aggregate class, we see that this time there is a significant difference in preferences. The single aggregate class prefers many more servers. This again is a consequence of the fact that in this case prioritization is a substitute for increasing the number of servers.

Column (b) of Figure 11 illustrates the same graphs for the case where the high priority jobs comprise less of the total load. The trends are the same as in column (a); however the preferred number of servers is significantly smaller in all figures. This follows from the same argument as that given for column (b) of Figure 10. In the case (not shown) where high priority jobs make up a greater proportion of the total load, the number of servers preferred is, as before, always higher than in column (a).

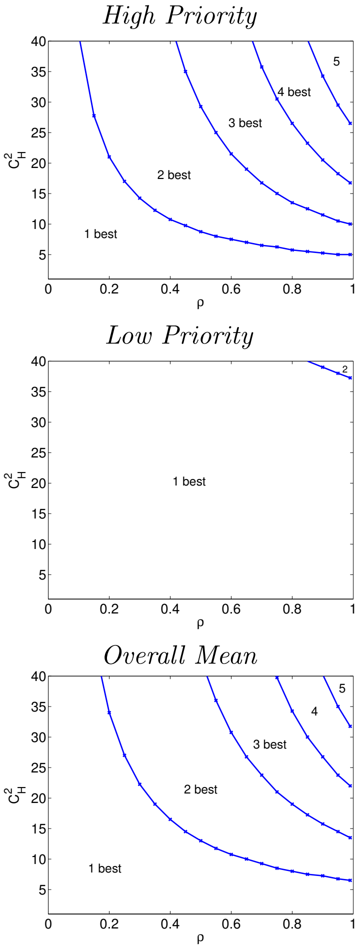
Figure 12: High priority class has larger mean

In Figure 12 column (a), we once again hold the mean high priority job size fixed at 1 and now assume the low priority job sizes have a mean size of 1/10. This case differs from the prior figure because now we are giving priority to the large job sizes: this reduces efficiency and, consequently, many more servers will be needed in this case.

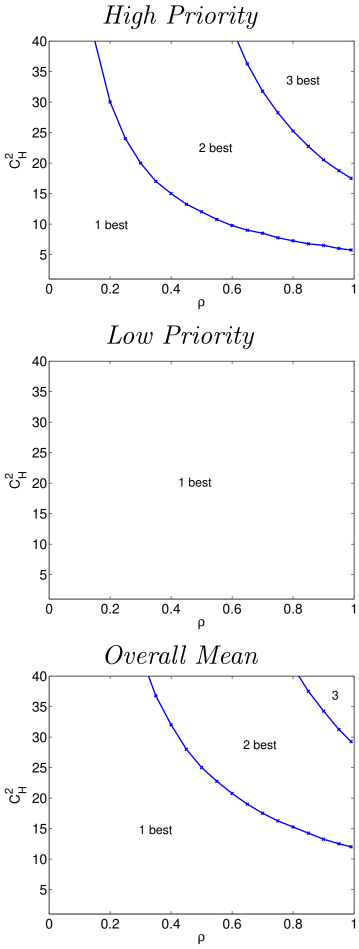
Once again, looking the topmost plot in column (a), we see that the preferred number of servers for high priority jobs is unaffected, since the high priority mean job size distribution has not changed. The low priority jobs, shown in the second plot of column (a), have vastly different preferences from the prior case. Here the low priority jobs prefer a very large number of servers; whereas in Figure 11 they almost always preferred one server. Because the low priority jobs are very small compared to the high priority jobs, they

(ii) $E[X_H] = 1$, $E[X_L] = 10$

2 Priority Classes

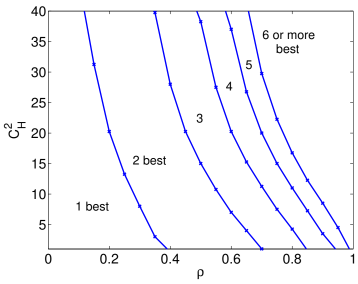


(a) $\rho_H = \rho_L$

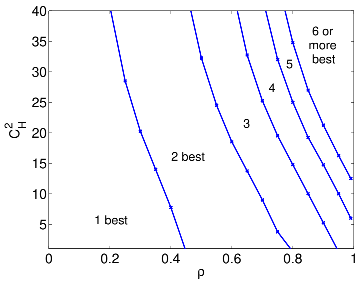


(b) $2\rho_H = \rho_L$

1 Aggregate Class



(a) $\rho_H = \rho_L$

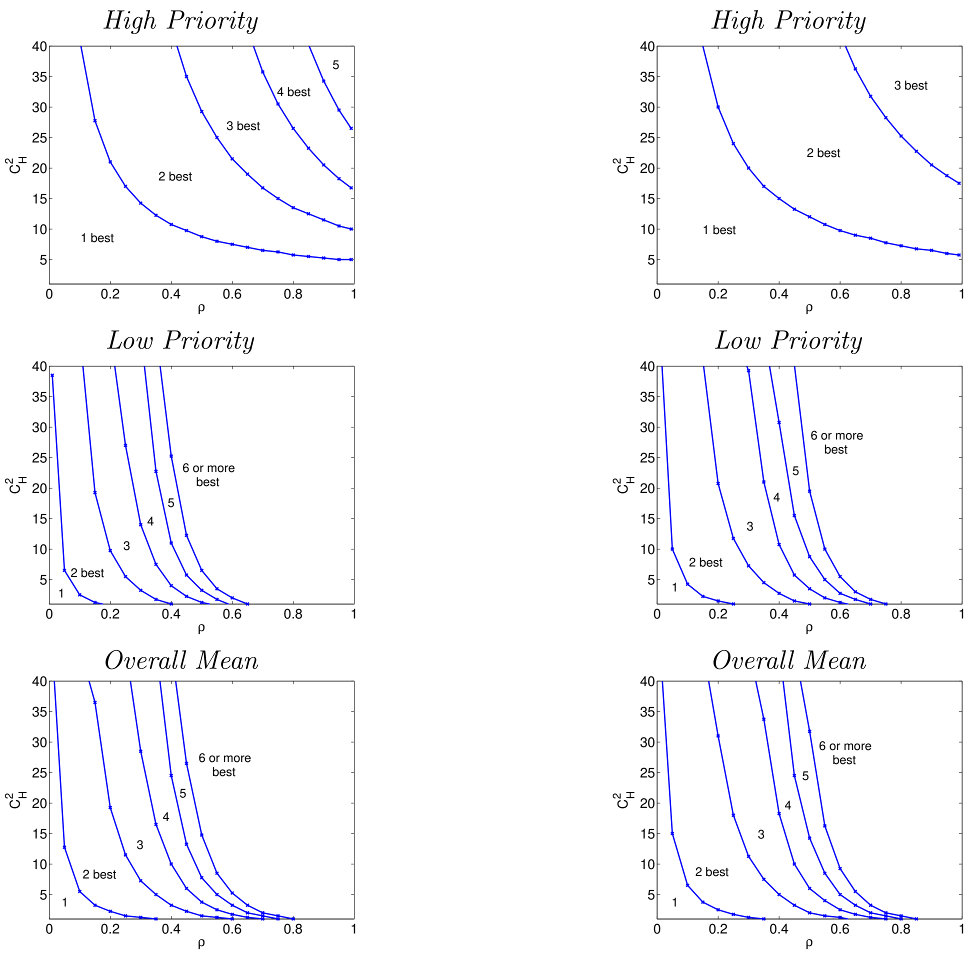


(b) $2\rho_H = \rho_L$

Figure 11: *How many servers are best (with respect to mean response time) when the high priority jobs have a smaller mean job size? Column (a) shows the case where the load made up by high priority jobs equals that of low priority jobs. Column (b) shows the case where the load made up by high priority jobs is half that of low priority jobs.*

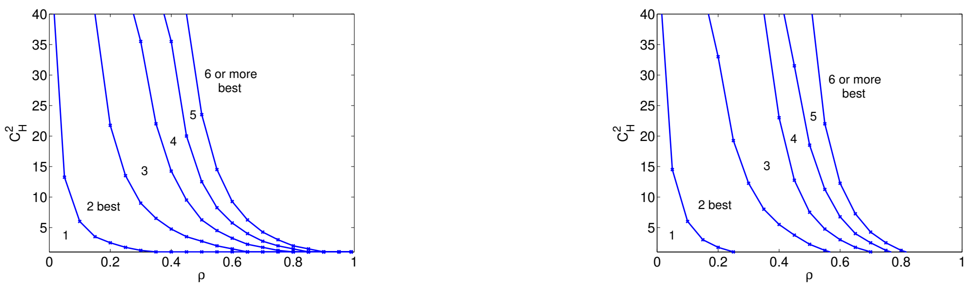
(iii) $E[X_H] = 1,\ E[X_L] = 1/10$

2 Priority Classes



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

1 Aggregate Class



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

Figure 12: *How many servers are best (with respect to mean response time) when the high priority class has a larger mean job size? Column (a) shows the case where the load made up by high priority jobs equals that of low priority jobs. Column (b) shows the case where the load made up by high priority jobs is half that of low priority jobs.*

want more servers in order to avoid being blocked, and forced to queue behind the large, high priority jobs.

The preferred number of servers for the overall mean response time in the dual-priority system, shown in the third plot of column (a), is again a hybrid of the preferences of the low and high priority jobs, but this time is strongly biased toward the low priority jobs because there are more of them. Notice therefore, that the number of servers preferred is much greater in this case. Comparing this with the single class aggregate, we see that the single class prefers slightly fewer servers than the dual class overall mean. This is due to the fact that the prioritization toward large jobs in the dual class system is inefficient (although despite it's inefficiency it may be mandated by external concerns). Note that because this case of prioritization is inefficient, the multiple servers provide an even larger benefit than in the other cases.

Column (b) of Figure 12 illustrates the same graphs for the case where the high priority jobs comprise less of the total load. The trends are the same as in Column (a); however the preferred number of servers is significantly smaller in all figures. This follows from the same argument as that given for column (b) of Figure 10. In the case (not shown) where high priority jobs make up a greater proportion of the total load, more servers are preferable.

Figure 13: Response time as a function of the number of servers

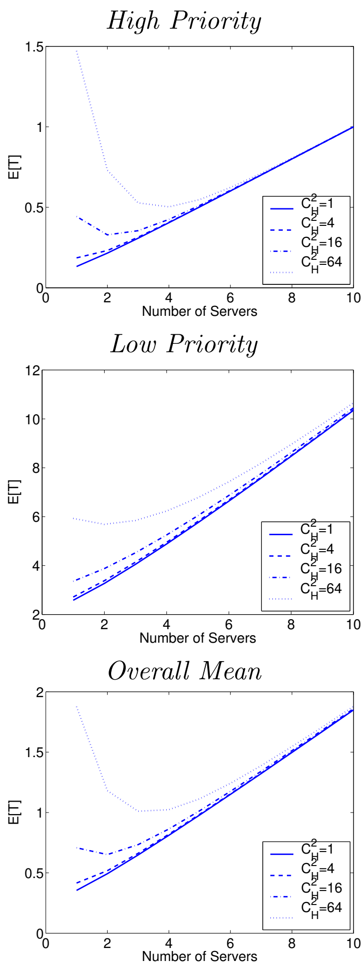
In all the prior results figures, we were concerned with determining the optimal number of servers as a function of system load and the variability of high priority jobs. Although we sometimes found k servers to be better than 1 server, we never looked at the actual mean response time as a function of the number of servers. In Figure 13 we do so, ranging the number of servers from 1 to 10. The key points made by this figure are that: (i) the mean response time of both priority classes is sensitive to the number of servers and (ii) increasing the number of servers may reduce mean response time up to a point; however making the number of servers too large increases mean response time – thus forming a "U-shape." This figure also reinforces the prior message that *the greater the variability of the high priority jobs, the greater the number of servers needed to mitigate this variability.*

Structurally, Figure 13 is divided into two columns: column (a) considers the job size distribution shown in Figure 11 and column (b) considers the distribution shown in Figure 12. In the previous figures, we have already discussed at length the differences in the number of servers preferred by each class. This same information can be read off of Figure 13 by observing that each of the plots in the figure have a "U-shape" and the bottom of the "U" indicates the optimal number of servers.

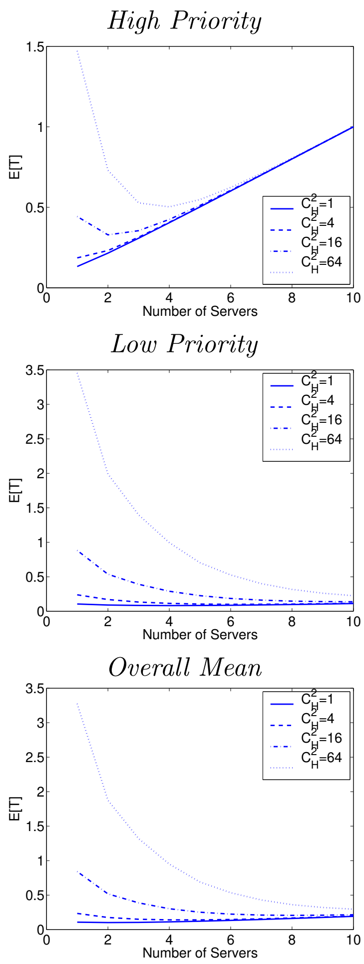
Figure 13 however makes the following additional points. First, we see that, under high variability ($C^2 = 64$), the difference in the overall mean response time between the case of 1 server and the optimal number of servers is about a factor of 2 in column (a) and, even more, close to a factor of 6 in column (b). Thus, variability does play a crucial role, imperfectly explored in prior research. Second, we see that, whereas in column (a) the optimal number of servers is quickly reached, in column (b) the optimal number of servers is in some cases greater than 10, not even appearing on the plot. Thus, how prioritization is performed has a large impact on how the system should be designed.

$\underline{\rho_H = \rho_L = 0.3}$

2 Priority Classes

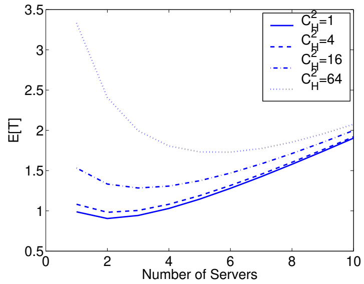


(a)$E[X_H] = 1,\ E[X_L] = 10$

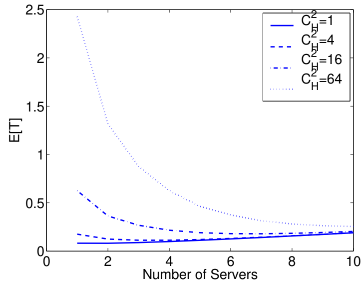


(b)$E[X_H] = 1,\ E[X_L] = \frac{1}{10}$

1 Aggregate Class



(a)$E[X_H] = 1,\ E[X_L] = 10$



(b)$E[X_H] = 1,\ E[X_L] = \frac{1}{10}$

Figure 13: *Mean response time as a function of the number of servers, which range from 1 to 10. The system load in these plots is $\rho = 0.6$. Column (a) shows the case where low priority jobs have a larger mean job size than high priority jobs. Column (b) shows the case where low priority jobs have a smaller mean job size than high priority jobs. (Note that the $E[X_H]$ and $E[X_L]$ listed are for $k = 10$ and are scaled appropriately for other k.)*

6 Summary and future work

We have presented the first accurate (within a few percent of simulation), computationally efficient analysis of the M/PH/k queue with dual priorities, which allows for phase-type service time distribution. Our method is conceptually simple: we transform a 2D-infinite Markov chain into a 1D-infinite Markov chain by leveraging QBD passage time results to compute the various types of busy period durations needed by our transitions. Furthermore, our method is fast – requiring less than one tenth of a second per data point in the figures of this paper. The speed of our method, combined with the fact that the method can handle PH service times, allows us to address many questions in the area of multiserver queues that have not been addressable in prior work.

In this paper we study the impact of server design (few fast servers versus more slow servers) on mean response time and per-class response time. We focus our analysis on answering the following four questions.

Under what conditions are multiple slow servers preferable to a single fast server? We find that for both the case of a single priority class and dual-priority classes, under high load and high service demand variability, more slow servers are preferable. Comparing Figures 10, 11, and 12, we see that the optimal number of servers for the dual-priority class can be lower than that for the single priority class when the low priority class has higher mean size than the higher priority class ($E[X_L] > E[X_H]$), and likewise can be higher than that for the single priority class when the high priority class has higher mean size than the low priority class ($E[X_H] > E[X_L]$).

Does the answer to "how many servers are optimal" differ for the different priority classes? We find that under dual-priority systems, the preferred number of servers turns out to be quite different when the goal is minimizing mean response time of just the high priority jobs versus minimizing the mean response time of just the low priority jobs. As seen across workloads (Figures 10, 11, and 12), the number of servers preferred by low priority jobs may be an order of magnitude higher, or lower, than that preferred by high priority jobs. For the case where priorities are chosen such that $E[X_L] > E[X_H]$, the number of servers preferred by low priority jobs is lower than that preferred by high priority jobs, and vice-versa when $E[X_H] > E[X_L]$.

How does the optimal number of servers in a dual-priority system differ from the case when all jobs have been aggregated into a single priority class? Aggregating the dual classes into a single priority class does *not* allow prediction of the best system configuration for minimizing mean response time, as shown in Figures 10, 11, and 12. If $E[X_L] < E[X_H]$, aggregation provides a reasonable approximation; however, when $E[X_L] > E[X_H]$ aggregation significantly mispredicts the optimal number of servers.

If one chooses a non-optimal number of servers, how does that affect mean response time and per-class response time? This question is answered in Figure 13, which shows that when job size variability is high, many slow servers are preferable to one fast server, and the performance difference may be a factor of $2-6$ in mean response time. However, even when job size variability is high, having too many slow servers can still be detrimental. Thus, there is an optimal server configuration that depends primarily on the variability of the job service demand and the system load.

With respect to *system design*, we have seen that choosing the correct number of servers can improve mean response time dramatically. Aside from improving mean response time, choosing the number of servers carefully can also ease the pain experienced by low priority jobs. We can mitigate the penalty to low priority jobs (particularly the penalty caused by high variability of high priority job sizes), by choosing a server configuration which is more favorable to low priority jobs. Figure 13 shows that this can often substantially improve the mean performance of low priority jobs without being detrimental to high priority jobs.

This entire paper has focussed on comparing one fast server versus k slow servers, each of $1/k$th the speed. Our reason for choosing this comparison is, first, the simplicity of comparing two systems with equal capacity, and second that prior work on this problem has also assumed equal capacity in comparing the two system configurations. However, it is also interesting to consider the effects of cost. In particular, k slow servers are typically far cheaper than a single fast server. If we wanted to compare two systems of equal cost, rather than equal service capacity, the same techniques presented in this paper could be used.

References

[1] A. Bondi and J. Buzen. The response times of priority classes under preemptive resume in M/G/m queues. In *ACM Sigmetrics*, pages 195–201, August 1984.

[2] L. Bright and P. Taylor. Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. *Stochastic Models*, 11:497–514, 1995.

[3] J. Buzen and A. Bondi. The response times of priority classes under preemptive resume in M/M/m queues. *Operations Research*, 31:456–465, 1983.

[4] J. Calabrese. Optimal workload allocation in open queueing networks in multiserver queues. *Management Science*, 38:1792–1802, 1992.

[5] X. Chao and C. Scott. Several results on the design of queueing systems. *Operations Research*, 48:965–970, 2000.

[6] R. Davis. Waiting-time distribution of a multi-server, priority queueing system. *Operations Research*, 14:133–136, 1966.

[7] W. Feng, M. Kowada, and K. Adachi. Analysis of a multiserver queue with two priority classes and (M,N)-threshold service schedule ii: preemptive priority. *Asia-Pacific Journal of Operations Research*, 18:101–124, 2001.

[8] H. Gail, S. Hantler, and B. Taylor. Analysis of a non-preemptive priority multiserver queue. *Advances in Applied Probability*, 20:852–879, 1988.

[9] H. Gail, S. Hantler, and B. Taylor. On a preemptive Markovian queues with multiple servers and two priority classes. *Mathematics of Operations Research*, 17:365–391, 1992.

[10] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. Squillante. Task assignment with cycle stealing under central queue. In *International Conference on Distributed Computing Systems*, pages 628–637, 2003.

[11] M. Harchol-Balter, T. Osogami, A. Scheller-Wolf, and A. Wierman. Multi-server queueing systems with multiple priority classes. *Queueing Systems*, 51, to appear.

[12] E. Kao and K. Narayanan. Computing steady-state probabilities of a nonpreemptive priority multiserver queue. *Journal on Computing*, 2:211–218, 1990.

[13] E. Kao and K. Narayanan. Modeling a multiprocessor system with preemptive priorities. *Management Science*, 2:185–97, 1991.

[14] E. Kao and S. Wilson. Analysis of nonpreemptive priority queues with multiple servers and two priority classes. *European Journal of Operational Research*, 118:181–193, 1999.

[15] A. Kapadia, M. Kazumi, and A. Mitchell. Analysis of a finite capacity nonpreemptive priority queue. *Computers and Operations Research*, 11:337–343, 1984.

[16] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* ASA-SIAM, Philadelphia, 1999.

[17] H. Leemans. *The Two-Class Two-Server Queue with Nonpreemptive Heterogeneous Priority Structures.* PhD thesis, K.U.Leuven, 1998.

[18] A. Mandelbaum and M. Reiman. On pooling in queueing networks. *Management Science*, 44:971–981, 1998.

[19] D. Miller. Steady-state algorithmic analysis of M/M/c two-priority queues with heterogeneous servers. In R. L. Disney and T. J. Ott, editors, *Applied probability - Computer science, The Interface, volume II*, pages 207–222. Birkhauser, 1992.

[20] I. Mitrani and P. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1:118–125, 1981.

[21] P. Molinero-Fernandez, K. Psounis, and B. Prabhakar. Systems with multiple servers under heavy-tailed workloads, 2003 – Manuscript.

[22] P. Morse. *Queues, Inventories, and Maintenance.* John Wiley and Sons, 1958.

[23] M. Neuts. Moment formulas for the Markov renewal branching process. *Advances in Applied Probabilities*, 8:690–711, 1978.

[24] B. Ngo and H. Lee. Analysis of a pre-emptive priority M/M/c model with two types of customers and restriction. *Electronics Letters*, 26:1190–1192, 1990.

[25] T. Nishida. Approximate analysis for heterogeneous multiprocessor systems with priority jobs. *Performance Evaluation*, 15:77–88, 1992.

[26] T. Osogami. *Analysis of multi-server systems via dimensionality reduction of Markov chains.* PhD thesis, Computer Science Department, Carnegie Mellon University, 2005.

[27] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. In *Performance TOOLS*, pages 200–217, 2003.

[28] A. Scheller-Wolf. Necessary and sufficient conditions for delay moments in FIFO multiserver queues with an application comparing s slow servers with one fast one. *Operations Research*, 51:748–758, 2003.

[29] G. Shahkar and H. Tareghian. Designing a production line through optimization of m/g/c using simulation. *Mathematical Engineering in Industry*, 8:123–126, 2001.

[30] A. Sleptchenko. Multi-class, multi-server queues with non-preemptive priorities. Technical Report 2003-016, EURANDOM, Eindhoven University of Technology, 2003.

[31] A. Sleptchenko, A. van Harten, and M. van der Heijden. An exact solution for the state probabilities of the multi-class, multi-server queue with preemptive priorities, 2003 – Manuscript.

[32] S. Stidham. On the optimality of single-server queueing systems. *Operations Research*, 18:708–732, 1970.