The Impact of Abandonment in Multi-class Priority Queues

Gwendolyn Stockman
Advisors: Mor Harchol-Balter and Adam Wierman

1 Introduction

Scheduling in web applications has been successful and is important in applications such as web servers and routers where policies other than first-come-first-served (FCFS) are often used. Often size-based prority scheduling policies are used and studied in practice. [12] studied DPS and [8] and [9] studied FB in routers. Lots of attention has been given to these policies in theory as well, see [2], [11]. In fact, a bias towards "small" jobs (in either age or size or remaining size) improves response time. And since user perceived performance is related to response time, efficiency is important. However, these policies which work to solve the problem of small jobs getting stuck behind large jobs, present the problem of users reneging and wasted processing time.

But in all of these, users are infinitely patient. This is not true in practice, where congestion leads to reneging. Studies found that a large portion of transfered data may be comprised of aborted transfers [1, 5, 13]. It was found that 11% of all transfers are interrupted, which corresponds to 20% of the transfer volume [5]. There are multiple reasons users may abort transfers; such as an incorrect file, poor performance, or a long setup time [13]. Poor performance during transfer may often be the result of bad policies for bandwidth sharing on a single bottleneck such as the link of a LAN which connects the LAN to the Internet [3].

Abandonment(user impatience) may cause problems for size-based scheduling. One of the biggest problems is wasted service, where service is given to jobs which later renege. In priority, class-based or size-based scheduling policies, service is often spread out over many jobs, so if a user grows impatient and reneges a job, that job may have received some work. This work is thrown away when the job abandons. Another type of scheduling is class-based. Where jobs are divided into classes based on size or priority. Work is then divided among the classes in some pre-determined way.

In many computer systems, priority queues, and size-based scheduling are used, however abandonment is often ignored. This occurs, because abandonment is hard to model. Though FCFS queues with abandonment can be analyzed [10], only limited work has been done outside of FCFS. Important work on abandonment in 2-class priority queues has been done previously (see Brandt and Brandt 2004, [4] and references therein), however this problem is not yet solved, and has not been generalized to the n-class system. *The goal of this paper is to present an effective approximation for the multi-priority FCFS queue, where jobs of high priority are served preemptively over jobs of lower priority, and jobs of equal priority are served in FCFS order.*

In order to develop an approximation for a multi-class priority queue, we first need an approximation for the $M/M/1-GI$ FCFS queue. We start by developing an approximation for the single-priority queue which allows abandonment after service has been recieved, and then analyze the multi-class priority queue.

The model we examine for the the single-priority case, is the $M/M/1-GI$ FCFS queue, where the first M indicates a Poisson arrival process, the second M represents an Exponential

service distribution, 1 server, and by the GI a general independent abandonment time distribution. Where jobs are scheduled according to FCFS scheduling. We assume that jobs know nothing about their position in the queue, and allow abandonment at the server. Note that this model has a waiting queue of infinite length. If this queue were to have finite length we would write $M/M/1/r - GI$ FCFS, where there are r waiting spaces in the queue. We will also discuss the $M/M/1 - M$ FCFS queue which is the same as the $M/M/1 - GI$ FCFS queue except that the abandonment time distribution is now Exponentially distributed, represented by the third M.

The model we examine is the multi-priority case is the n-priority $M/M/1 - GI$ queue, where there are n priority classes, each priority class with their own queue as above. We assume that jobs are preemptible. While we are only looking at an Exponential service distribution, we think that the service can be extended to phase-type distributions.

The main focus of this paper is the problem of non-exponential abandonment time distributions in multi-class priority queues. Our goal is to present an effective approximation for calculating performance metrics in the $M/M/1 - GI$ queue. While we only look at percent abandoning we believe our approximation and analysis can be extended to calculate any of the other metrics discussed in [10].

In Section 2 we describe and evaluate our approximation for the $M/M/1 - GI$ FCFS queue. In Section 3 we develop an approximation of the multi-priority $M/M/1 - GI$ system, using busy periods and our approximation from Section 2. And in Section 4 we present our conclusions and future work.

2 The $M/M/1 - GI$ FCFS Approximation

In order to approximate a multi-class priority queue with general abandonment, we first need a good approximation of a single priority FCFS queue with general abandonment. In this section we generalize the approach by Whitt in [10] in order to allow abandonment at the server. In Section 2.1 we present background on the analysis of an $M/M/1 - M$ FCFS queue. In Sections 2.2 through 2.5 we develope the approximation. We then validate and discuss this approximation in Section 2.6.

2.1 M/M/1-M FCFS Queue

We start with the simple case of exponential abandonment in order to illustrate the analytic approach. If the abandonment times have an Exponential distribution, instead of a general independent distribution, then we have the much simpler model, the $M/M/1 - M$ FCFS, which can be analyzed exactly. Note, when we talk about state k, we mean a total of k jobs in the system, 1 at the server and $k - 1$ in the queue.

To work this out we draw a Markov chain and calculate the steady-state distribution (the π_i's) using local balance equations. The steady state distribution can then be used to calculate a variety of user metrics (e.g. mean response time, percent abandonment). The fact that we can draw a Markov chain for the $M/M/1 - M$ FCFS case is very important to our analysis, because we will approximate the $M/M/1 - GI$ queue with a similar Markov chain.

However, note that, approximating the $M/M/1 - GI$ queue with a $M/M/1 - M$ queue where the abandonment is exponentially distributed having mean equal to that of the general distribution is not good enough. Thus, we need to model more than just the mean of the general distribution.

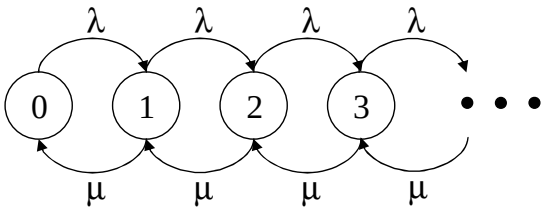


Figure 1: A Markov Chain with arrival rate λ and service rate μ. The number in each circle represents the number of jobs in the system.

2.2 $M/M/1-GI$ FCFS Approximation

We now generalize the simple analysis of the $M/M/1-M$ FCFS queue in order to approximate the $M/M/1-GI$ FCFS queue. Our approach mimics that of [10], but extends the analyss to allow jobs to abandon while recieving service. This extension is key to allowing us to use the approximation to model a priority queue. Let the rate of abandonment for a job at time t be the time-to-abandon failure rate (hazard) function

$$h(t) = \frac{f(t)}{\bar{F}(t)}, \qquad t \geq 0, \tag{1}$$

where $f(t)$, and $\bar{F}(t)$ are the density and the complementary cdf respectively associated with the time-to-abandon function $F(t)$.

Note that the abandonment rate of a job depends on the time it has been waiting. However, this is not tracked in our Markov Chain, so we need to estimate this quantity. To estimate this assume that the state of a job is the length of the queue and the job's position in the queue. Assume that the server is busy, and that there are k jobs waiting in the queue. If there were no abandonments then the job j^{th} from the end of the queue would have seen exactly $j-1$ arrivals while waiting. If we assume that abandonments are relatively rare then, estimate that there were j new arrival events since that job entered the queue. We will see that this approximation works even under high abandonment.

2.3 Markovian Approximation

We will use the above observation in order to approximate the $M/M/1-GI$ FCFS queue using a Markov chain. In order to draw a Markov chain for the $M/M/1-GI$ FCFS queue, we need an approximation for the death rate from a state of the Markov chain. We now develope such an approximation. Note, when we talk about state k, we mean a total of k jobs in the system, 1 at the server and $k-1$ in the queue. We let the states $1, 2, \ldots$ represent the number of jobs in the system (i.e. the number of jobs at the server + the number of jobs in the queue). The arrival rate at each state is λ.

In order to determine the total death rate from a state of the Markov chain, we need to determine the rate of abandonment from that state. To account for the non-constant hazard rates (either increasing or decreasing) we apply a delta-approximation to the hazard rate curve. We estimate the time between arrivals to be $1/\lambda$, so if there were exactly j arrivals since the job j^{th} from the end of the queue arrived, this job has been in the queue for approximately j/λ time. Define the abandonment rate of the j^{th} job from the end of the queue as:

$$\alpha_j \equiv \frac{h((j-1)/\lambda) + h(j/\lambda)}{2}, \qquad 1 \leq j \leq k+1. \tag{2}$$

When calculating the total abandonment from the queue with k jobs in the queue and one at the server is, we include the abandonment rate of the job at the server in addition to the abandonment rate of each job waiting in the queue. So, the total abandonment from the system with k is:

$$\delta_k \equiv \sum_{j=1}^{k} \alpha_j = \sum_{j=1}^{k} \frac{h((j-1)/\lambda) + h(j/\lambda)}{2}. \tag{3}$$

Since there is abandonment in every state (even when there is only 1 job in the system, the one at the server), we have that the total death rate from state k is:

$$\mu_k = \mu + \delta_k \qquad 1 \leq k \tag{4}$$

2.4 Solve for the π_is

Let $\pi_k = P$(there are k jobs in the system (# at server + # in queue). Thus $\sum_{i=0}^{\infty} \pi_i = 1$. We can solve for these using the local balance equations:

$$\pi_k \lambda = \pi_{k+1} \mu_{k+1}, \qquad 0 \leq k \leq r \tag{5}$$

When we solve for them we get:

$$\pi_0 = \frac{1}{\sum_{i=0}^{\infty} \frac{\lambda^i}{\prod_{k=1}^{i} \mu_k}} = \frac{1}{\sum_{i=0}^{\infty} \prod_{k=1}^{i} \frac{\lambda}{\mu_k}} \tag{6}$$

$$\pi_i = \frac{\lambda^i}{\prod_{k=1}^{i} \mu_k} \pi_0 = \pi_0 \prod_{k=1}^{i} \frac{\lambda}{\mu_k} \tag{7}$$

2.5 Probability of Abandoning

In general let p_k^a be the probability that there are k jobs in the system when an arrival enters the system. We let,

$$p_k^a = \pi_k \qquad \forall k \geq 0. \tag{8}$$

Consider a job arriving when there are k jobs in the system, 1 at the server and $k-1$ waiting in the queue. After this job's arrival there will be $k+1$ in jobs in the system. Following the argument in [10] we consider the system starting at $k+1$ jobs in the system ignoring future arrivals. Figure 2.5 describes what the system looks like at this point.

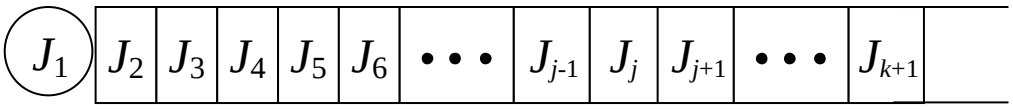


Figure 2: Job J_1 is at the server and jobs $J_2, \ldots, J_{k+1}$ are waiting in the queue. Each job J_i has abandonment rate $\alpha_{k+1-(i-1)} = \alpha_{k+2-i}$ for $1 \leq i \leq k+1$.

Now, we need a way to estimate the time between departures. While departures are not necessarily rare, we make the assumption that all departures up to a certain point were completions. Thus we approximate the time between successive departures as $1/\lambda$. With time between successive departures estimated at $1/\lambda$, after d successive departures, d/λ time was passed. So

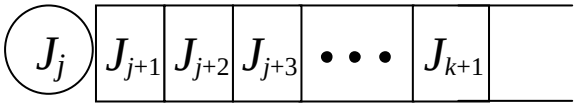


Figure 3: Job J_j is at the server and jobs $J_{j+1}, \ldots, J_{k+1}$ are waiting in the queue. Each job J_i has abandonment rate $\alpha_{k+2-i+(j-1)} = \alpha_{k+1+j-i}$ for $j \leq i \leq k+1$.

the job originally j^{th} from the end of the queue, if still present, originally had abandonment rate $\alpha_j = \frac{h(\frac{j-1}{\lambda})+h(\frac{j}{\lambda})}{2}$, but now has abandonment rate $\alpha_{j+d} = \frac{h(\frac{j-1}{\lambda}+\frac{d}{\lambda})+h(\frac{j}{\lambda}+\frac{d}{\lambda})}{2} = \frac{h(\frac{j+d-1}{\lambda})+h(\frac{j+d}{\lambda})}{2}$.

Let $\gamma_{k,j} = P($the job initially k^{th} in line abandons in the j^{th} departure event given that that job has not yet abandoned), and let $m_{k,j}$ be the average time between the $(j-1)^{\text{st}}$ and the j^{th} departure events.

After $j-1$ departures, using the assumption that the departures were all completions, the first $j-1$ jobs in the queue $(J_2, J_3, \ldots, J_j)$ have gone into service, while the last $k-j+1$ jobs $(J_{j+1}, J_{j+2}, \ldots, J_{k+1})$ are still waiting, see Figure 2.5. Also, the job that was originally at the server and the first $j-2$ jobs of the $j-1$ jobs which went into service (jobs $J_1, \ldots, J_{j-1}$), have completed service and left the system, and job J_j is at the server.

The job originally at the server (J_1) had abandonment rate α_{k+1} and has completed. Also, the first $j-2$ jobs in the original queue (jobs $J_2, \ldots, J_{j-1}$) have completed service, and their original abandonment rates were

$$\alpha_{k+1-((j-1)-1)} = \alpha_{k-(j-2-1)} = \alpha_{k-j+3}, \ldots, \alpha_k = \alpha_{k+1-(2-1)}.$$

The job which is currently at the server (job J_j), originally had abandonment rate $\alpha_{k-(j-2)} = \alpha_{k-j+2}$, and now has abandonment rate $\alpha_{k+1+j-j} = \alpha_{k+1}$ because $j-1$ departures, and we are assuming that the average time between departures is $1/\lambda$, so $\frac{j-1}{\lambda}$ time has passed. The remaining $k-j+1$ jobs in the queue (jobs $J_{j+1}, \ldots, J_{k+1}$), are still in the queue and their abandonment rates increased from $\alpha_1, \ldots, \alpha_{k-(j-1)} = \alpha_{k-j+1}$ to $\alpha_{1+(j-1)} = \alpha_j \ldots, \alpha_{k+1+j-(j+1)} = \alpha_k$.

So the total remaining abandonment rate is

$$\alpha_{k+1} + \sum_{i=j}^{k} \alpha_i = \sum_{i=1}^{k+1} \alpha_i - \sum_{i=1}^{j-1} \alpha_i = \delta_{k+1} - \delta_{j-1}, \tag{9}$$

and the total remaining departure rate is $\mu + \delta_{k+1} - \delta_{j-1}$.

This gives us that the probablity the job initially k^{th} in line abandons in the j^{th} departure event given that that job has not yet abandoned is

$$\gamma_{k,j} = \frac{\alpha_j}{\mu + \delta_{k+1} - \delta_{j-1}} \tag{10}$$

and the average time between the $(j-1)^{\text{st}}$ and the j^{th} departure events is

$$m_{k,j} = \frac{1}{\mu + \delta_{k+1} - \delta_{j-1}} \tag{11}$$

where $1 \leq j \leq k+1$ and $\delta_0 = 0$.

Note that these probabilities are the same for the system approximated in [10] where jobs at the server do not abandon, if we let there be initially k jobs in the queue ($k+1$ jobs in the system), when an arrival occurs, and the analysis begins. (So $k+2$ jobs in the system after the arrival).

The probability that the job originally k^{th} in the queue eventually receives service is

$$\Gamma_k^r = (1-\gamma_{k,1})(1-\gamma_{k,2})\cdots(1-\gamma_{k,k}) = \prod_{i=1}^{k}(1-\gamma_{k,i}). \tag{12}$$

Similarly, the probability that the job originally k^{th} in the queue eventually completes service is

$$\begin{aligned}
\Gamma_k^c &= P(\text{the job originally } k^{\text{th}} \text{ in the queue completes service}) \cdot \Gamma_k^r \\
&= \frac{\mu}{\mu+\alpha_{k+1}}\Gamma_k^r = \frac{\mu+\alpha_{k+1}-\alpha_{k+1}}{\mu+\alpha_{k+1}}\Gamma_k^r = \left(1-\frac{\alpha_{k+1}}{\mu+\alpha_{k+1}}\right)\Gamma_k^r \\
&= \left(1-\frac{\alpha_{k+1}}{\mu+\sum_{i=1}^{k+1}\alpha_i-\sum_{i=1}^{k}\alpha_i}\right)\Gamma_k^r = \left(1-\frac{\alpha_{k+1}}{\mu+\delta_{k+1}-\delta_k}\right)\Gamma_k^r \\
&= \left(1-\frac{\alpha_{k+1}}{\mu+\delta_{k+1}-\delta_{(k+1)-1}}\right)\Gamma_k^r = \left(1-\gamma_{k,(k+1)}\right)\Gamma_k^r \\
&= \prod_{i=1}^{k+1}(1-\gamma_{k,i})
\end{aligned} \tag{13}$$

Also, we must compute not only the probability that a job that enters the system eventually receives service, but also the probability that a job which enters the system eventually completes service. To do this we need to look at the probability that a job arriving to the empty system will complete service before abandoning.

It follows from above, that the probability that a job who enters the system eventually receives service is

$$P(S^r) = p_0^a + \sum_{k=0}^{\infty} p_{k+1}^a \Gamma_{k+1}^r \tag{14}$$

and that the probability that a job which enters the system eventually completes service is

$$P(S^c) = p_0^a P(X < A) + \sum_{k=0}^{\infty} p_{k+1}^a \Gamma_{k+1}^c. \tag{15}$$

where $P(X < A)$ is the probability that a job arriving to the empty system will complete service before abandoning.

So the probability that a job abandons given that it entered the system is

$$P(A) = 1 - P(S^c). \tag{16}$$

Summary of Approximation for the $M/M/1-GI$ FCFS Queue:
We approximate and $M/M/1-GI$ queue with arrival rate λ, service rate μ, and abandonment distribution with hazard rate $h(t)$ and mean $E[A]$. To do this, first recall from equations (2), (3) that

$$\alpha_j = \frac{h((j-1)/\lambda) + h(j/\lambda)}{2}, \qquad 1 \le j \le k+1$$

and

$$\delta_k \equiv \sum_{j=1}^{k+1}\alpha_j = \sum_{j=1}^{k+1}\frac{h((j-1)/\lambda) + h(j/\lambda)}{2}.$$

Using equations (6) and (7) we get

$$\pi_i = \frac{\prod_{k=1}^{i} \frac{\lambda}{\mu_k}}{\sum_{i=0}^{\infty} \frac{\lambda^i}{\prod_{k=1}^{i} \mu_k}}$$

further, we have

$$\Gamma_k^c = \prod_{i=1}^{k+1} \left(1 - \frac{\alpha_i}{\mu + \delta_{k+1} - \delta_{i-1}}\right)$$

and

$$P(S^c) = p_0^a P(X < A) + \sum_{k=0}^{\infty} p_{k+1}^a \Gamma_{k+1}^c.$$

from equations(13), (10), and(15). Note that $p_i^a = \pi_i$.

Note that the two summations in these formulas need to be truncated, and for any truncation the error can be easily understood. Also note that we believe that this approximation is extendable to ther metrics.

Remark 1 *The main change from the approximation in [10] is to allow abandonment at the server, which results in several specific changes overall. First, to calculate the total abandonment from a state of the queue, we must include the rate of abandonment from the job at the server. In order to to calculate the probability that a job abandons given that it entered the system, we need to know the probability that it completed service not just that the job in question recieved service. So we must calculate $P(S^c)$ not just $P(S^r)$, and thus we get $P(A) = 1-P(S^c)$ not $P(A) = 1 - P(S^r)$.*

Also, we need to estimate $P(X < A)$, that is the probability that a job arriving to the empty system will complete service before abandoning, in order to calculate the the probability that a job which enters the empty system completes service, and thus, the probability that a job which enters the system at all completes service. Another change caused by allowing abandonment at the server, is the need to use a delta-approximation instead of just taking the right endpoint of the interval of the hazard rate curve. This is needed to account for the non-constant hazard rates (either increasing or decreasing), especially for cases where there is a very steep increasing or decreasing portion of the hazard rate curve, for example the very steep decreasing hazard rate at the beginning of a Weibull curve. Another change is the total remaining abandonment as seen in equation (9), where we once more must include the abandonment of the job currently at the server, not just the abandonment of the jobs in the queue.

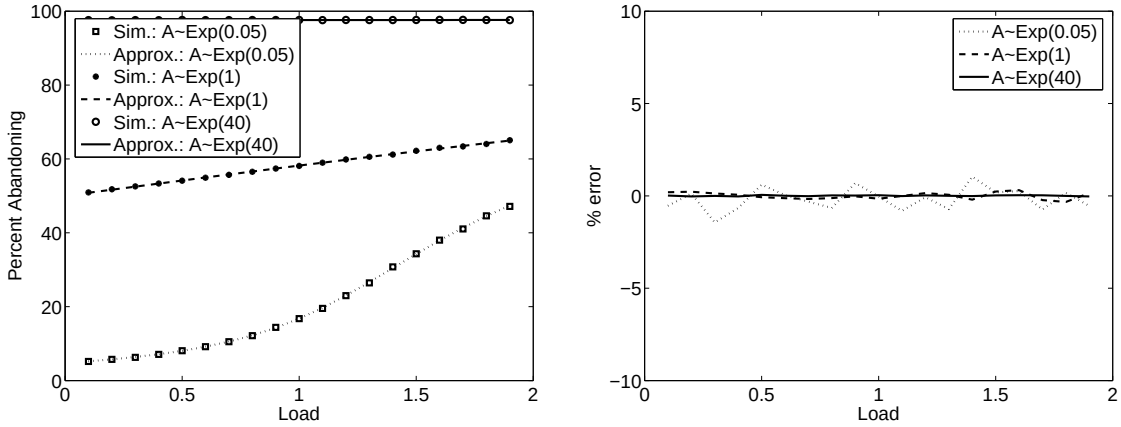
Remark 2 *This approximation would work also in the case of a finite queue, with at most r waiting spaces, where only several changes would be necessary.*

1. *In equations (3) and (4) the finite queue case, k is bounded by the number of jobs allowed in the system.*

2. *The sums in equations (6), (14), and (15) include only a finite number of terms.*

3. *Also, in equation 8 we need to know the probability that a job is rejected from the system when it arrives. This occurs if there are $r + 1$ jobs in the system when it arrives (r in the queue, and 1 at the server), so the probability that a job is rejected is*

$$P(reject) = \pi_{r+1}. \tag{17}$$



(a) Percent Abandoning

(b) Absolute error

Figure 4: Percent abandoning for the $M/M/1-M$ queue, with $E[X]=1$, for several different abandonment distributions. For $A \sim Exp(0.05)$, $A \sim Exp(1)$, $A \sim Exp(40)$, we have $\mathbb{E}[A]=20$, $\mathbb{E}[A]=1$, and $\mathbb{E}[A]=.025$, respectively, corresponding to low, medium, and high abandonment

and

$$p_k^a = \frac{\pi_k}{1-P(reject)} = \frac{\pi_k}{1-\pi_{m+1}} \qquad \text{for } 0 \le k \le r+1, \tag{18}$$

2.6 Validation and Discussion

Before moving to the analysis of priority queues, we will first validate the FCFS approximation and investigate the impact of non-exponential abandonment distributions. We start with the simplest case: the $M/M/1-M$. Note that in this case we can calculate the π_i's, $\gamma_{k,j}$'s exactly and therefore Γ_k^c's exactly. This implies that every term in the sum to get $P(S^c)$ is exact, also $P(X<A)$ and therefore $P(S^c)$ is exact.

Therfore we can use this to validate our simulator and to understand baseline simulation error. The simulations were run for 3 iterations each with a total of 100,000 jobs entering the system. The results of these simulations where then averaged. As seen in figure 2.6 the error is very close, in fact it is within 2% at all times, which validates our similator.

We now move to validating our approximation under general abandonment distributions. We look at the Weibull which can have increasing, constant, and decreasing hazard rates. The distribution for the Weibull(a,b) is

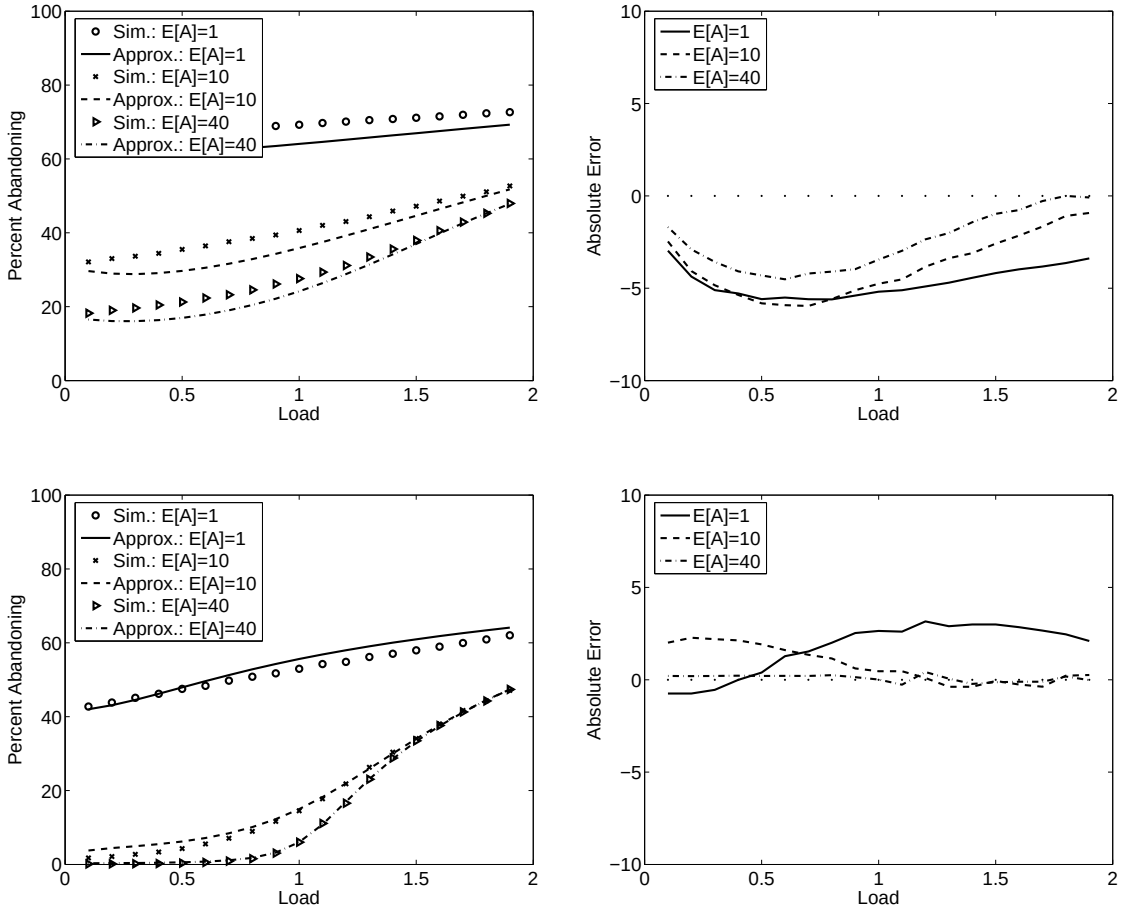
$$F(x) = 1 - e^{-(\frac{t}{a})^b}$$

and the hazard rate function is

$$h(t) = b\frac{t^{b-1}}{a^b}.$$

We focus on 3 cases:

1. Increasing hazard rate with $b=2$
2. Constant hazard rate, which was already discussed, because is equivalent to an Exponential hazard rate.
3. Decreasing hazard rate with $b=0.5$



(a) Percent Abandoning (b) Absolute error

Figure 5: In the top row we look at the case of a decreasing time-to-abandon function. The means of 1, 10, and 40 correspond to Abandonment $\sim$ Wei$(0.5, 0.5)$, Wei$(5, 0.5)$, and Wei$(20, 0.5)$ respectively. In the bottom row we have an increasing time-to-abandon function. The means of 1, 10, and 40 correspond to Abandonment $\sim$ Wei$(\frac{2}{\sqrt{\pi}}, 2)$, Wei$(\frac{20}{\sqrt{\pi}}, 2)$, and Wei$(\frac{80}{\sqrt{\pi}}, 2)$ respectively.

We fixed the mean across comparisons. For each case we let the mean of the abandonment vary across $1, 10$, and 40, which corresponds to high, medium, and low abandonment situations.

A large part of the error is from the fact that we are not constantly updating the rate of abandonment, we only do this when a job arrives or departs. We can see this in Figure 2.6 where we underestimate for the most part in the case of a decreasing hazard rate, and we overestimated for the most part in the case of an increasing time-to-abandon function. These errors are due in large part to the shape of the hazard curve of the Weibulls.

We believe that this approximation would work in the $M/GI/1{-}GI$ case, by approximating the service distribution by an exponential service distribution with an equal mean, so if the service distribution has mean $\mathbb{E}[X]$ then we would set $\mu = \frac{1}{\mathbb{E}[X]}$. However, more work is necessary to verify this.

3 Priority Queue Approximation

We now build on our $M/M/1-G$ FCFS approximation in order to approximate the performance of $M/M/1-G$ priority queues. Tthe n-priority queue is makde up of n seperate queues, labeled $0, 1, \ldots, n-1$, where jobs of priority i go straight into priority queue i. Within each queue jobs are served in FCFS order, and jobs in queue i are served only when the queues labeled $0, 1, \ldots, i-1$ are empty.

Note that priority 0 jobs experience a FCFS system with arrival rate λ_0, and the approximation from Section 2 applies immediately. However, for jobs of lower priority we need to use more information. In order to determine the time a job j of priority i has been waiting, we need to take into account not only the jobs in priority queue i in front of the job in question, but also the jobs in the higher priority queues both when the job, j, enters the system, and those high-priority jobs which arrive while the job is in the system. The key idea of our approximation is to approximate the experienced service distribution of these lower priority jobs using busy periods. In Section 3.1 we explain the general setup and assumptions we are making about the priority queue. Next in Section 3.2 we develope the approximation. Finally, in Section 3.3 we validate and discuss this approximation.

3.1 The Setup

Assume that there is a priority queue with n priority classes, labeled $0, 1, \ldots, n-1$, where the highest priority class is labeled 0 and the lowest priority class is labeled $n-1$. So a job of priority class i has greater priority than a job of priority class $i+1$. When we refer to priority queue i we mean the queue of jobs of priority class i. Also, assume that jobs enter the system with a priority of $0, 1, \ldots, n-1$ already assigned. Jobs in priority class i enter the system with rate λ_i. So the total rate of jobs entering the system is $\lambda = \sum_{i=0}^{n-1} \lambda_i$.

We assume the n priority classes have service distributions with means $\mathbb{E}[X_0], \mathbb{E}[X_1], \ldots \mathbb{E}[X_{n-1}]$, loads $\rho_0, \rho_1, \ldots, \rho_{n-1}$, and abandonment distributions with time-to-abandon hazard functions $h_0(t), h_1(t), \ldots, h_{n-1}(t)$, and means $\mathbb{E}[A_0], \mathbb{E}[A_1], \ldots, \mathbb{E}[A_{n-1}]$, respectively.

Let

$$\rho = \sum_{i=0}^{n-1} \rho_i \tag{19}$$

and let q_i be the probability that a job entering the system is of class i, so the arrival rate of jobs with priority i is $\lambda_i = \lambda q_i$, and $\rho_i = \lambda q_i \mathbb{E}[X_i]$. Now for $0 \leq i \leq n-1$, $\rho_i = \lambda q_i \mathbb{E}[X_i]$ implies

$$\lambda = \frac{\rho_i}{q_i \mathbb{E}[X_i]} \tag{20}$$

and

$$q_i = \frac{\rho_i}{\lambda \mathbb{E}[X_i]}. \tag{21}$$

Note that $\sum_{i=0}^{n-1} q_i = 1$, so

$$1 = \sum_{i=0}^{n-1} q_i = \sum_{i=0}^{n-1} \frac{\rho_i}{\lambda \mathbb{E}[X_i]} = \frac{1}{\lambda} \sum_{i=0}^{n-1} \frac{\rho_i}{\mathbb{E}[X_i]}$$

$$\lambda = \sum_{i=0}^{n-1} \frac{\rho_i}{\mathbb{E}[X_i]} \tag{22}$$

And by equation (20) we have

$$\frac{\rho_j}{q_j \mathbb{E}[X_j]} = \lambda = \sum_{i=0}^{n-1} \frac{\rho_i}{\mathbb{E}[X_i]}$$

so,

$$q_j = \frac{\rho_j}{\mathbb{E}[X_j] \sum_{i=0}^{n-1} \frac{\rho_i}{\mathbb{E}[X_i]}} \qquad \text{for } 0 \leq j \leq n-1. \tag{23}$$

3.2 The Approximation for priority class i

In this section we show how we use busy periods to approximate the mean of the experienced service distribution of priority i jobs, which we will call $\mathbb{E}[\widetilde{X}_i]$. Using this estimated experienced service distribution of priority i jobs, we use the approximation for the single priority $M/M/1-GI$ FCFS queue, see the summary of the approximation in Section 2.5, with arrival rate λ_i and service distribution with mean $\mathbb{E}[\widetilde{X}_i]$ to approximate user metrics.

3.2.1 How it works

Clearly, the behavior of jobs in priority class 0 can be approximated by the $M/M/1-GI$ FCFS approximation given above, with arrival rate λ_0, service distribution with mean $E[X_0]$, and time-to-abandon function $h_0(t)$. However, for jobs in priority classes i for $i > 0$, this is not the case, because they may have to wait while jobs of higher priority classes are served.

A priority i job waits behind k other jobs of the same priority. Each job can only complete when the system is empty of higher priority jobs, so the time for a low priority job to complete is a busy period composed of jobs of classes higher than i, started by a job in class i.

So for priority classes $i > 0$, we estimate the experienced service time of a job of that class using a busy period started by a job of class i, containing jobs of higher priority classes. By definition of a busy period, see [6] and [7], we see that

$$\mathbb{E}\begin{bmatrix}\text{length of a busy period of jobs of priority} \\ \text{higher than } i\text{, started by a job of priority } i\end{bmatrix} = \frac{\mathbb{E}[X_i]}{1-\rho_{i-1}}. \tag{24}$$

However, since jobs of the higher priority classes also experience abandonment, the actual load experienced by a priority queue j, for $0 \leq j < i$, is less then the load calcualted by just using the arrival rate and the service rate. To account for this, we estimate the experienced load of queue j, call it ρ_j^*. Let $\widetilde{\rho}_i = \lambda_i \mathbb{E}[\widetilde{X}_i]$, and $\widetilde{\rho}_i^* = 1 - \widetilde{\pi}_{0,i}$ where $\widetilde{\pi}_{0,i}$ is π_0 for the single priority $M/M/1-GI$ FCFS queue with arrival rate λ_i service distribution with mean $\mathbb{E}[\widetilde{X}_i]$ and abandonment distribution with mean $\mathbb{E}[A_i]$. The idea behind using an estimate for the experienced load of a queue i, instead of the value calculated using the arrival and service distributions, is that abandonment in a queue decreases the experienced load. And since load is defined to be the proportion of time that the server is busy, the experienced load is $1-$ the proportion of time there are no jobs at the queue. This lowers our estimate of $\mathbb{E}[\widetilde{X}_i]$ and makes it more accurate.

To calculate user metrics for priority class i, use the approximation for the single priority $M/M/1-GI$ FCFS queue with arrival rate λ_i, service distribution with mean $\mathbb{E}[\widetilde{X}_i]$ and abandonment distribution with mean $\mathbb{E}[A_i]$.

$$\mathbb{E}[\widetilde{X}_i] = \begin{cases} \mathbb{E}[X_i] & i = 0 \\ \frac{\mathbb{E}[X_i]}{1-\widetilde{\rho}_{i-1}^*} & 1 \leq i \leq n-1. \end{cases}$$

The overall percent abandoning is just $\sum_{i=0}^{n-1} q_i$(percent abandoning from class i).

3.3 Validation and Discussion

We now move to validating the priority queue qpproximation and illustrating the effect of abandonment in realistic senarios. We start with the simple case of exponential abandonment in all priority classes and then move to the more realistic settings.

In the simple case of Exponential abandonment in all priority classes the only error results from the busy period approximation, since the approximation is exact for priority class 0 and if the busy period estimation of the experienced service distribution of the priority class i jobs was exact, then the estimate for the jobs of priority i would be exact, for $i > 0$. For the non-exponential abandonment case, the error also comes from the $M/M/1 - M$ FCFS approximation.

We tested this approximation in the 2-priority cases. We ran several different cases

1. $\mathbb{E}[X_i] = \mathbb{E}[X_j]$ and $E[A_i] = E[A_j]$ $\forall i, j \in [0, n-1]$
2. $\mathbb{E}[X_i] < \mathbb{E}[X_j]$ for $i < j$, $i, j \in [0, n-1]$
   (a) $E[A_i] = E[A_j]$ for $i < j$, $i, j \in [0, n-1]$
   (b) $E[A_i] < E[A_j]$ for $i < j$, $i, j \in [0, n-1]$
   (c) $E[A_i] > E[A_j]$ for $i < j$, $i, j \in [0, n-1]$

In each case we look at low, medium, and high abandonment. We look at all cases for completeness but case, 2b is the most pratical because the smaller higher priority jobs have a smaller mean abandonment time, which approximates the case where abandonment is proportional to size. Also, this is the case we are best at. The simplest case is case 1 where everything is the same. Another simple case is 2a where although the means of the service distributions are different, the means of the abandonment distributions are equal. Finally, we thought we would do badly in case 2c, but we do well.

In our simulations we ran 3 iterations over 100,000 arrivals, and averaged the results. We will call jobs of priority class 0, high priority jobs, and jobs of class 1, low priority jobs. We let $\mathbb{E}[X_0] = 0.1$ and $\mathbb{E}[X_1] = 1$. We ran simulations for all combinations of $\mathbb{E}[A_0] = 1, 10, 20$ and $\mathbb{E}[A_1] = 1, 10, 20$.

The most pratical setting is when the low priority jobs really do see a busy period, this is seen when $\mathbb{E}[X_0] = 0.1$ and $\mathbb{E}[X_1] = 1$, and $\mathbb{E}[A_0] = 1$ and $\mathbb{E}[A_1] = 20$, which is shown in Figure 3.3 and 3.3. In fact this is the case where we are the most accurate. Note that capturing the busy period approximation is not very important in this setting because, high priority jobs are very small, and therefore they have a low impact on the experienced service time of low priority jobs. Also, high priority jobs are not abandoning much. This is a realistic setting and leads to situations where our approximation is accurate. It is natural to think that there are setting where the approximation is inaccurate, however these are mainly non-realistic settings.

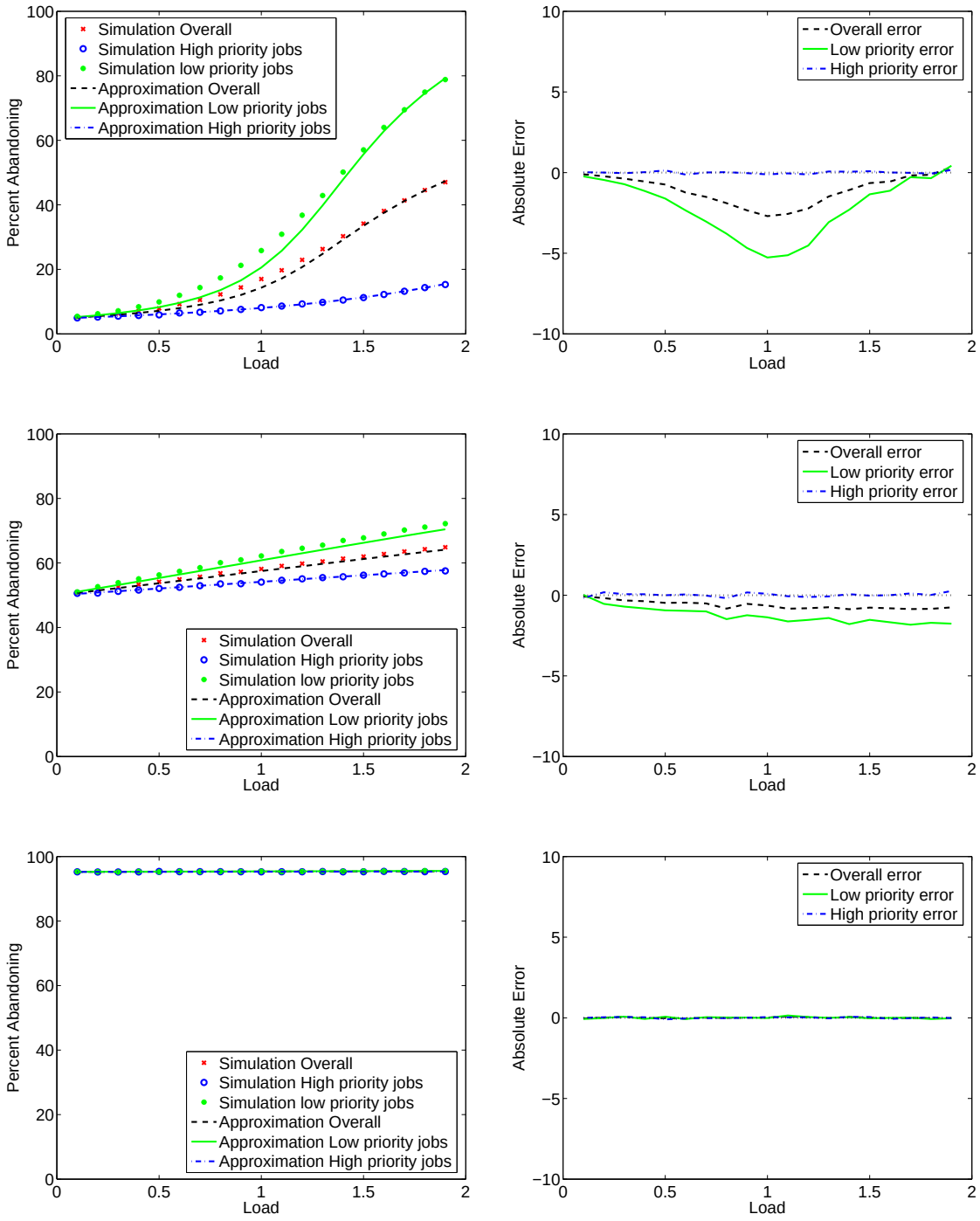
Note that our absolute error stays within a margin of 2 in all but one case where it is within a margin of 6. Also we are almost always underestimating the percent abandoning, however the error does not seem to increase with load.

Note that the most abandonment is in the case where $\mathbb{E}[X_H] = 0.1$ and $\mathbb{E}[X_L] = 1$ and $\mathbb{E}[A_L] = \mathbb{E}[A_H]$.

4 Conclusion

We give a new approximation for priority queues, that is simple and accurate, especially in realistic settings. The key idea of the approximation is the discretizing of the general independent abandonment times, through the hazard rate function of the time-to-abandon

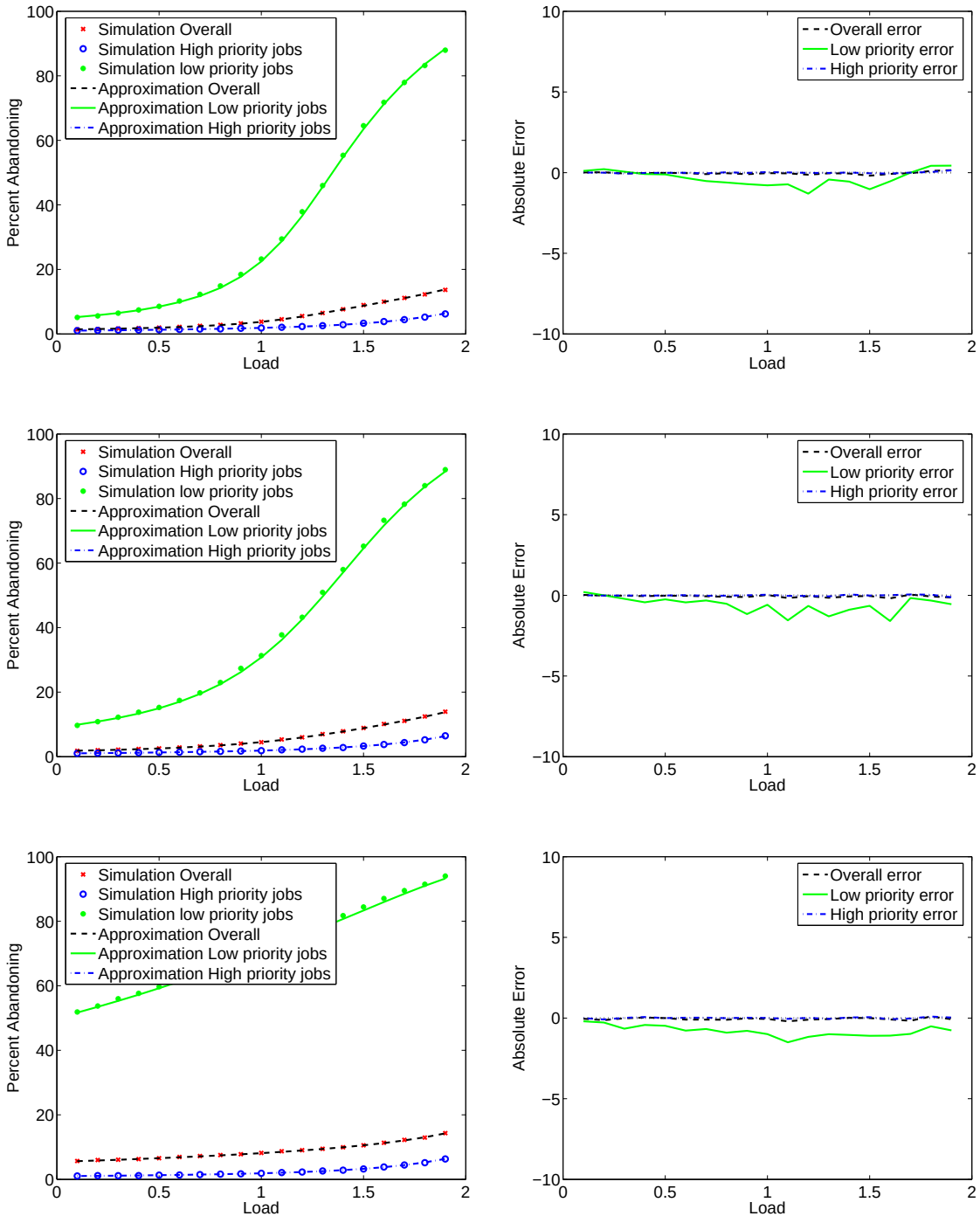
1. $\mathbb{E}[X_0] = \mathbb{E}[X_1] = 1$ and $E[A_0] = E[A_1]$



(a) Percent Abandoning (b) Absolute error

Figure 6: Row 1 shows the case where $E[A_0] = E[A_1] = 20$, that is AbandonmentẼxp(0.05). Row 2 shows the case where $E[A_0] = E[A_1] = 1$, that is AbandonmentẼxp(1). Row 3 shows the case where $E[A_0] = E[A_1] = 0.05$, that is AbandonmentẼxp(20). Also note that $\rho_0 = \rho_1$.

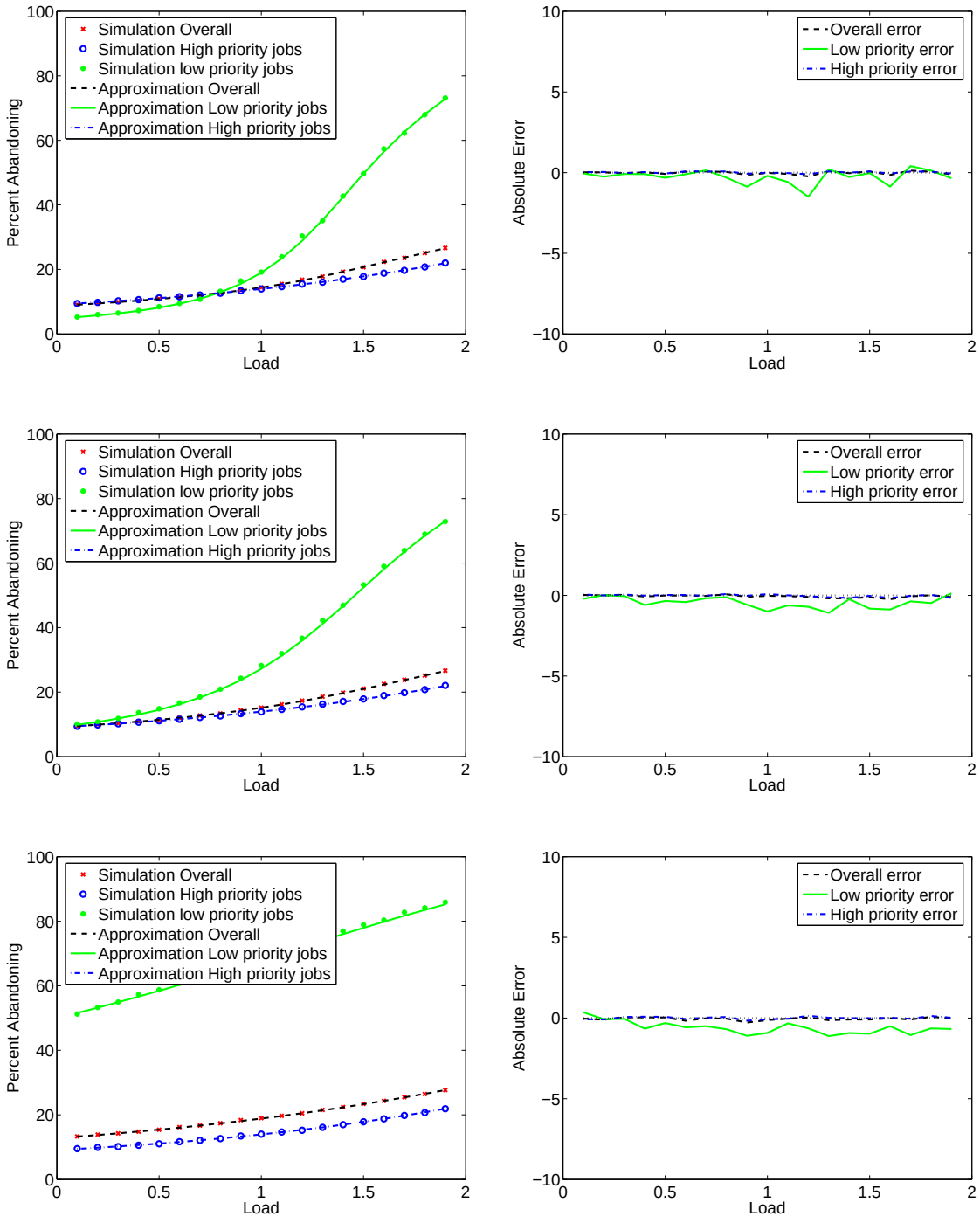
2. $\mathbb{E}[X_0] = 0.1$, $\mathbb{E}[X_1] = 1$ and $E[A_0] = 10$



(a) Percent Abandoning (b) Absolute error

Figure 7: Row 1 shows case 2b where $E[A_1] = 20$, that is $A_1\tilde{\text{E}}\text{xp}(0.05)$. Row 2 shows the case 2a where $E[A_1] = 10$, that is $A_1\tilde{\text{E}}\text{xp}(0.1)$. Row 3 shows the case 2c where $E[A_1] = 1$, that is $A_1\tilde{\text{E}}\text{xp}(1)$. Note that $\rho_0 = \rho_1$.

2. $\mathbb{E}[X_0] = 0.1$, $\mathbb{E}[X_1] = 1$ and $E[A_0] = 1$



(a) Percent Abandoning (b) Absolute error

Figure 8: Row 1 shows case 2b where $E[A_1] = 20$, that is $A_1\tilde{\text{E}}\text{xp}(0.05)$. Row 2 shows the case 2b where $E[A_1] = 10$, that is $A_1\tilde{\text{E}}\text{xp}(0.1)$. Row 3 shows the case 2a where $E[A_1] = 1$, that is $A_1\tilde{\text{E}}\text{xp}(1)$. Note that $\rho_0 = \rho_1$.

distribution. This allows the abandonment times to be used in a queue with time-varying service. We use this discretization to approximate the $M/M/1-GI$ FCFS queue, then we build on it to approximate a multi-class priority queue. The key idea in the priority queue approximation is to use busy periods to capture the interaction between high and low priority jobs.

We have performed computer simulations to validate the performance of these approximations, and the examples we have considered cover a wide variety of scenarios. The absolute error between the simulation and approximation is within 6 everywhere, and within 2 in most cases, including the most realistic settings. In fact our approximation works best in the most realistic settings.

While we are only examining the user metric of percent abandoning, we believe that our approximation can be extended by using our approximation in to the equations in [10]. Another important note is that we believe our approximation can be extended to the $M/GI/1-GI$ case, however more work is necessary to check this. Future work will also include testing the n-priority approximation in the case where $\mathbb{E}[X_i] > \mathbb{E}[X_j]$ for $i < j$, and verifying the approximation with more priority classes.

References

[1] M. Arlitt and C. Williamson. Web server workload characterization: the search for invariants. In *Proc. of ACM Sigmetrics*, 1996.

[2] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. In *Proceedings of ACM Sigmetrics*, 2001.

[3] J. Boyer, F. Guillemin, P. Robert, and B. Zwart. Heavy tailed M/G/1-PS queues with impatience and admission control in packet networks. In *IEEE Infocom*, 2003.

[4] A. Brandt and M. Brandt. On the two-class M/M/1 system under preemptive resume and impatience of prioritized customers. *Queueing Systems*, 47, 2004.

[5] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of web proxy caching in a heterogeneous bandwidth environment. In *Proc. of IEEE INFOCOM*, 1999.

[6] L. Kleinrock. *Queueing Systems*, volume I. Theory. John Wiley & Sons, 1975.

[7] L. Kleinrock. *Queueing Systems*, volume II. Computer Applications. John Wiley & Sons, 1976.

[8] I. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *Proceedings of ACM Sigmetrics*, 2003.

[9] I. A. Rai, G. Urvoy-Keller, M. Vernon, and E. W. Biersack. Performance modeling of LAS based scheduling in packet switched networks. In *Proc. of ACM Sigmetrics-Performance*, 2004.

[10] W. Whitt. Engineering solution of a basic call-center model. *Management Science*, 51(2):221–235, Feb 2005.

[11] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proceedings of ACM Sigmetrics*, 2003.

[12] S. Yang and G. de Veciana. Size-based adaptive bandwidth allocation: optimizing the average qos for elastic flows. In *Proc. of IEEE INFOCOM*, 2002.

[13] S. Yang and G. de Vecianca. Bandwidth sharing: the role of user impatience. In *Proc of Globecom*, 2001.