Noname manuscript No.
(will be inserted by the editor)

Many flows asymptotics for SMART scheduling policies*

Changwoo Yang · Adam Wierman · Sanjay Shakkottai · Mor Harchol-Balter.

Received: date / Accepted: date

Abstract Scheduling policies that favor small jobs have received growing attention due to their superior performance with respect to mean delay, e.g., Shortest Remaining Processing Time (SRPT) and Preemptive Shortest Job First (PSJF). In this paper, we study the delay distribution of the class of scheduling policies called SMART-LD, which includes SRPT, PSJF, and a range of practical variants, in a discrete-time queueing system under the many sources large deviations regime.

Our analysis of SMART-LD in this regime (large number of flows and large capacity) hinges on a novel two dimensional queueing framework that employs virtual queues and total ordering of jobs. We prove that all SMART-LD policies have the same asymptotic delay distribution as SRPT, i.e., the delay distribution has the same decay rate. In addition, we illustrate the improvements SMART-LD policies make over First Come First Serve (FCFS) and Processor Sharing (PS).

Our two-dimensional queueing technique is generalizable to other policies as well. As an example, we show how the Foreground-Background (FB) policy can be analyzed using a two-dimensional queueing framework. FB is a policy, not contained in SMART-LD, which manages to bias towards small jobs without knowing which jobs are small in advance.

Keywords many sources large deviation · two dimensional queueing · Shortest remaining processing time (SRPT) · Foreground-background (FB)

* A preliminary version of this paper appeared in the Sigmetrics 2006 conference proceedings. A discussion of the differences between the two papers is included in the cover letter.

Changwoo Yang
11304 Vist Sorrento Pkwy 200. San Diego, CA 92130, USA
E-mail: sylvant@gmail.com

Adam Wierman
Computer Science at the California Institute of Technology, 1200 E. California Blvd. Pasadena, CA 91125
E-mail: adamw@caltech.edu

Sanjay Shakkottai
ECE department at The University of Texas at Austin
E-mail: shakkott@ece.utexas.edu

Mor Harchol-Balter
Computer Science at Carnegie Mellon University, 5000 Forbes Ave. Pittsburgh, PA 15213
E-mail: harchol@cs.cmu.edu

1 Introduction

Increasingly, computer system designers are replacing traditional schedulers with schedulers that favor small (short) requests. Instead of using designs based on policies such as First Come First Served (FCFS) and Processor Sharing (PS), which shares the service capacity evenly among all requests in the system, designs are increasingly using schedulers modeled after Shortest Remaining Processing Time (SRPT). For example, traditional web server designs share bandwidth evenly among all open connections, while recent improvements suggest using variations of SRPT to schedule bandwidth to connections [9, 24, 13, 28]. In addition, traditional router designs share bandwidth evenly among flows, while recent improvements suggest favoring short flows by giving priority to those flows which have sent the fewest packets so far, in accordance with the Foreground-Background (FB) policy [22, 23, 8]. The same pattern has been repeated in wireless networks [10], peer to peer systems [21], and beyond.

The reason for this shift towards schedulers that favor small jobs is simple. SRPT has long been known to minimize mean delay and mean queue length [27]. However, system designers are interested in more than just minimizing mean delay and mean queue length; the distribution of delay is also very important, particularly in providing Quality of Service guarantees. Unfortunately, the analysis of the delay distribution under non-FCFS scheduling policies is known to be difficult. As a result, the study of the delay distribution is focused on asymptotics. In particular, two asymptotic frameworks are considered: (i) the *large buffer large deviations framework* and (ii) the *many sources large deviations framework*.

Prior work studying the delay distribution of scheduling policies has predominantly been in the large buffer framework, which studies the likelihood of large delays, W, specifically, $Pr(W > y)$ as $y \to \infty$. By contrast, current paper provides new results characterizing the delay distribution of a range of scheduling policies in the many sources framework. The many sources framework scales the number of arrival flows (N), the buffer size (B), and the service capacity (C), proportionally, as shown in Figure 1. In this regime, the focus is on deriving the asymptotic *decay rate* of delay. In particular, we are interested in the asymptotic decay rate of the distribution of delay experienced by a job of size k, $W^{(N)}(k)$, as $N \to \infty$, i.e., $Pr(W^{(N)}(k) > m)$ for any finite m as $N \to \infty$. We will provide a formal definition in Section 2.

Both the large buffer and many sources large deviations frameworks provide important information about the delay distribution of scheduling policies. However, our focus on the many sources framework in this paper is motivated by applications such as high traffic web servers and routers that have enormous available bandwidth and thousands of simultaneous flows. Further, notice that the many sources framework studies $W(k)$, the delay for jobs of size k, instead of W, the delay of a job of any size. By providing the delay for jobs of different sizes, through the many source framework, we can study why and how the distribution of delay of large jobs "starve" under policies that favor small jobs. This is an important extension since, to this point, the unfairness experienced by large jobs has tended to be studied in expectation, e.g., [31, 22, 23, 33].

While there is a large literature studying scheduling policies in the large buffer regime, see [14, 20, 19] and the references therein for a survey; until recently there were very few results in the many sources regime. In particular, only the decay rates of FCFS [4], a simple priority queueing system [7, 30], and Generalized Processor Sharing [11] were known. Then, recently, Yang & Shakkottai [36] derived the decay rate of SRPT. *In this paper, we extend this literature by deriving the decay rate of the SMART classification and FB, which we motivate below.*

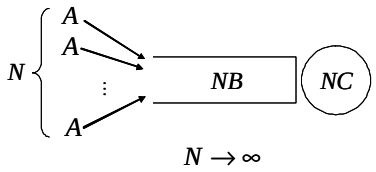


Fig. 1 In the many sources framework, the capacity and the buffer size of a system are scaled proportionally with the number of sources. As shown in the figure, we consider the delay probability of far away from the mean when the system is accessed by a large number of sources.

In recent years, researchers have begun to focus on a class of policies that formalizes the heuristic of "prioritizing small jobs" termed the SMART class, e.g., [34,33,32,19]. This stream of research is motivated by the fact that computer system designs almost never implement the idealized policies studied in theory, e.g., SRPT, but rather implement variations motivated by the heuristic underlying SRPT, e.g., "prioritize small jobs." The SMART class includes a wide range of scheduling policies that follow this heuristic of favoring small jobs. The analysis of the SMART classification provides both practical and theoretical benefits when compared with studying SRPT alone. Practically speaking, the study of the SMART class provides analytic results for the policies that are actually implemented in computer systems; and theoretically, such results add structure to the space of scheduling policies that cannot be obtained by analyzing individual policies alone. Prior to this paper, it has been shown that all SMART policies have mean delay within a factor of 2 of optimal in the M/GI/1 [34] and that all SMART policies have an asymptotically equivalent delay distribution in the large buffer regime [19]. However, SMART policies have not been studied in the many sources regime.

In addition to studying the SMART classification, we study an important related policy that is not included in the SMART class – the Foreground-Background (FB) policy.[1] FB scheduling is "blind" to job-size information, but attempts to prioritize small jobs by using the attained service (age) of a job as an indication of its remaining size. FB is known to minimize mean delay among blind policies when job sizes have a decreasing failure rate (DFR). The delay distribution of FB has been characterized in the large buffer large deviations regime [19], but has not been studied in the many sources regime. We will provide more detail on FB in Section 5, and the interested reader can refer to [18] for a survey on FB.

This paper makes three main contributions to the understanding of SMART and FB. First, we define a classification that generalizes SMART called SMART-LD (Definition 1) and we prove that all SMART-LD policies have the same asymptotic decay rate (Theorem 1). Thus, the practical variations of SRPT included in SMART-LD are all equivalent to SRPT with respect to delay in the many sources regime. Second, we derive the asymptotic decay rate of FB (Theorem 2). It is important to point out that the decay rate of FB differs from that of SMART-LD, which is why we could not generalize SMART-LD to include FB. We validate the derived decay rate of SMART-LD and FB using simulations. In particular, we illustrate that the distribution of $W(k)$ converges to the many sources asymptote very quickly: convergence is achieved after only 20 flows. Considering that high traffic web servers and routers routinely have many more than 20 simultaneous flows, this provides practical moti-

[1] FB is known in the literature under a variety of other names including: Least Attained Service first (LAS) and Shortest-Elapsed-Time first (SET).

vation for the many sources scaling. Finally, we perform an extensive analytic and numerical comparison of the asymptotic decay rates of SMART-LD, FB, FCFS, and PS. Further, we use numerical calculations to compare the decay rate of $W^{(N)}(k)$ under SMART-LD, FB, PS, and FCFS across load, job size, job size distribution, and delay threshold (m). This is an important comparison given that PS and FCFS are often the status-quo in computer systems. Lastly, we prove that all SMART-LD policies stochastically outperform FB with respect to the distribution of delay for all job sizes (Corollary 1), and investigate the magnitude of this improvement using numerical calculations. This comparison illustrates the price FB pays for not using job size information to schedule.

Our results are enabled through the use of a new analytic framework that we refer to as the *two dimensional queueing framework*. This framework adds tie-break rules to policies in a way that does not alter the asymptotic performance of the policies, but greatly simplifies their analysis (see Section 3). By adding tie-break rules, the jobs in the system are sufficiently ordered to allow clean separation of high and low priority jobs with respect to the job in question thus making the analysis feasible. The strength of this novel framework is that it enables: *(i)* the study of policies that depend on the job state (age and/or remaining size), as opposed to only the queue length; and *(ii)* the first study of a *class of policies* in the many sources regime, as opposed to only the analysis of individual policies.

The paper is organized as follows. In Section 2, we will formally introduce our model and the many sources regime. Then, in Section 3, we will give an overview of the two dimensional queueing framework that is the key to our analysis of both SMART-LD and FB. Next, in Section 4, we introduce the SMART-LD classification and prove that all SMART-LD policies are asymptotically equivalent in the many sources regime. Then, in Section 5, we introduce the FB policy and derive its decay rate. Having analyzed both SMART-LD and FB, we next compare these policies with each other, and with FCFS and PS in Section 6. Finally, we conclude in Section 7.

2 System Setup

In this section, we introduce the many sources large deviations framework for which the delay analysis of SMART-LD and FB is carried out and state the basic assumptions made in this paper. We consider a queueing system with a single queue and a single server having stationary and ergodic arrival and service processes, where the arrival and service processes are independent of each other. The system operates in discrete time, i.e. a batch of jobs arrive at the beginning of each time slot and jobs are serviced at the end of each time slot. The queue state is measured immediately after the service and just before the arrivals of the next time slot.

In the many sources regime, the number of arrival processes is scaled along with the capacity of the system and the buffer size as depicted in Figure 1. We assume that the possible sizes of jobs are restricted to bounded multiples of a unit size. Thus, we represent the set of possible job sizes as $\mathcal{M} = \{1, 2, 3, \ldots, M\}$, where M is the largest job size. The assumption that the service distribution is bounded is natural given the numerous recent studies that have observed that file sizes at web servers typically follow a bounded, highly variable distribution size [1,5]. Formally, for each job size $k \in \mathcal{M}$, we assume N independent, identically distributed arrival processes. We define $A^N(a, b)$ as the total number of arrivals by all N arrival processes in the (a, b) time-interval[2], where $a \leq b$. For example, $A^N(0, 0)$ signifies

[2] The notation (a, b) refers to time slots $\{a, a + 1, \ldots, b\}$.

the total number of arrivals in time slot 0. Additionally, we define $A_k^N(a,b)$ as the total number of jobs of size k that arrive in the queue during time-interval (a,b). Thus, the volume of size k arrivals is $kA_k^N(a,b)$, and $A^N(a,b) = \sum_{k=1}^{M} A_k^N(a,b)$. We assume independence between arrival processes of different sized jobs, i.e., $A_i^N(a,b)$ is independent of $A_j^N(a,b)$ for $i \neq j$. Note that job arrivals from a single stream of any given size can be correlated across time-slots.

As depicted in Figure 1, we assume that the capacity of the server, C, is scaled in proportion to the number of arrival processes, and at most NC units of work can be serviced at any time slot. We assume that the server is work-conserving and that the system is stable, i.e. $E\left[\sum_{i=1}^{M} iA_i^N(0,0)\right] < NC$.

Our goal is to study the tail probability of delay in the many sources regime. The delay, $W^{(N)}(k)$, is the delay experienced by the last job of size k in an arrival burst to a stationary system. The tail probability of delay, $\Pr(W^{(N)}(k) > m)$, is the probability that the last job of size k in the burst arriving at time slot l does not leave the system by the end of time slot $l+m$. It has been shown that, in the large deviation framework, the tail probability of delay of various scheduling policies such as FCFS, Generalized Processor Sharing (GPS), and Priority Queueing decays as

$$\Pr(W^{(N)}(k) \geq m) = g^N(k,m)e^{-NI_W(k,m)},$$

under general conditions, where $g^N(k,m)$ is a function that satisfies $\lim_{N\to\infty} -\frac{1}{N}\log g^N(k,m) = 0$. In other words, the most dominant trend of the tail probability is the exponential decay $I_W(k,m)$, which is appropriately called the *decay rate* [29]. Thus, the decay rate of delay is defined as

$$I_W(k,m) = \lim_{N\to\infty} -\frac{1}{N}\log \Pr(W^{(N)}(k) > m). \tag{1}$$

In this paper, we show that such decay rates for SMART-LD and FB exist, and we derive their precise form. Note that the delay distribution for a job of size k depends on the capacity C, the threshold value m, the job size k, and the arrival processes $A_k^N(a,b)$ $\forall k \in \mathcal{M}$. In particular, the contribution of the arrival process to the delay decay rate of scheduling policies is expressed through their own decay rate, which has been well analyzed in literature [29], i.e.,

$$I_{A_k}^{(a,b)}(x) = \lim_{N\to\infty} -\frac{1}{N}\log \Pr(A_k^N(a,b) > Nx). \tag{2}$$

Our goal is to study the decay rate of delay. To do so, we first consider the distribution of the *virtual delay,* and then relate this quantity back to the delay distribution. The *virtual delay*, $V^{(N)}(k)$, is the delay seen by a fictitious (virtual) job that arrives at Q_k, the queue for size k jobs, at the end of an arrival burst at $t = 0$ (given that the system started at $t = -\infty$). The event $\left\{V^{(N)}(k) > m\right\}$ corresponds to a fictitious job arriving at the end of an arrival burst during time slot 0 and not departing the system until the mth time slot. Note that this setup ensures that the system is stationary at the arrival of the virtual job. To avoid confusion, we will refer to the delay $W^{(N)}(k)$ as the *actual delay* in order to distinguish it from the *virtual delay* $V^{(N)}(k)$. Observe that the virtual delay is different from actual delay: for example, even when there is no arrival, the virtual delay can be measured, whereas the actual delay is not defined. The decay rate of the virtual delay [30] is defined as

$$I_V(k,m) = \lim_{N\to\infty} -\frac{1}{N}\log \Pr(V^{(N)}(k) > m). \tag{3}$$

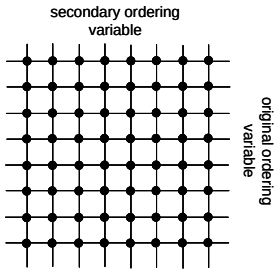


Fig. 2 Two Dimensional Queueing Framework. Although, we consider a single queue we make use of an array of virtual queues, depicted as dots, arranged in a two dimensional grid to facilitate analysis of various scheduling policies. Each virtual queues correspond to jobs with distinct primary and secondary ordering variables.

3 Two dimensional queueing framework

In order to analyze the SMART-LD class and FB in the many sources large deviations regime, we develop a new analytic framework that we refer to as the *two dimensional queueing framework* and denote it by 2DQ framework for short. The proposed 2DQ framework makes use of a collection of virtual queues, that contains all jobs in a specific state, arranged in a two dimensional grid as shown in Figure 2. The virtual queues are defined and arranged so that all relevant states of the system at any time can be easily represented by the 2DQ framework.

The state of a system can be represented by the two dimensional collection of virtual queues (2DQ) due to the fact that the relevant job state is *discrete*. This is due to the following two assumptions made in Section 2: (i) The job sizes are restricted to multiples of a unit size, (ii) The server serves jobs in discrete amounts. The 2DQ framework allows the delay analysis of SMART-LD and FB by providing a coherent and consistent portrayal of the system state, which changes as jobs receive service, based on the following two main concepts: *finiteness* and *ordering*.

First, by *finiteness*, the number of virtual queues in 2DQ required to fully represent the state of the system is finite, which makes the analysis much simpler. We have assumed that the server services the jobs in *discrete* amounts and that the set of possible job sizes is $\mathcal{M} = \{1, 2, 3, \ldots, M\}$. The important consequence of this is that at any time slot, the distribution of all relevant states of any job in SMART-LD and FB (original size, remaining size, and attained service) is *discrete* and *finite*. This results in a more tractable description of the queue state.

The second important concept of the 2DQ framework is *ordering*. Many scheduling policies specify a scheme of ordering (prioritization), in which the server considers some jobs more important than others and serves those first. For example, FCFS orders jobs by their time of arrival, SRPT by the remaining size (i.e., remaining service requirement), and FB by their age (i.e., attained service). Note that when all jobs in the system are totally ordered, the analysis becomes feasible for many scheduling policies. In other words, the concept of ordering should be taken one step further by assigning a further ordering scheme, beyond the existing one, to specify total ordering or the lack of it. For example, SRPT does not specify any rule for jobs with the same remaining size; thus some additional ordering should be defined to determine which of the jobs with equal remaining size should be served first. Based on this observation, the 2DQ framework takes the concept of ordering inherent

in the given scheduling policy one step further by assigning a secondary ordering scheme to the inherently existing one as depicted in Figure 2. In other words, jobs are serviced in the order specified by the scheduling policy, but when there are multiple jobs with the same priority, the secondary ordering scheme is used to select the next job to serve. For example, let us consider the ordering of "smaller job size first" as the secondary ordering for FCFS. In this case, when multiple jobs arrive to the system at the same time slot, smaller jobs can be served before larger jobs. The importance of the secondary ordering is that it further constrains the policy, thus making the analysis more tractable, as we will see in the cases of SMART-LD and FB.

We note that by applying the 2DQ framework for the analysis of the policies in the SMART-LD class and FB, we *are not altering the performance of these policies in the asymptotic framework*. The finiteness and ordering in the 2DQ framework are simply modeling aids. In particular, we will use job size as the secondary ordering variable for FB and SMART-LD. However, this does not mean that FB needs to know the size of a job; rather, using the job size as the secondary ordering criteria makes the analysis of FB more tractable.

4 The SMART-LD class

It is well known that policies that "give priority to small jobs" perform well with respect to mean delay. As we have already discussed, this idea has been fundamental to many computer systems applications ranging from web servers and routers to supercomputing centers and operating systems. However, although the same idea of favoring small jobs guides all these implementations, the specific scheduling policies that result differ due to (i) implementation restrictions and (ii) concerns about metrics other than mean delay (e.g., avoiding starvation of large jobs). In particular, variations of SRPT are used instead of the idealized SRPT that is studied in the literature. For example, whereas SRPT assumes a continuum of possible priority levels, in practice coarser implementations of SRPT are used, and these only a small finite number (e.g., 5) of priority levels. Another example is that a practical version of SRPT might combine PS and SRPT so as to attain the delay benefits of SRPT while maintaining some of the fairness benefits of PS.

Recently, the SMART class was introduced to capture a wide range of practical variations of SRPT, all based on the common heuristic of "giving priority to small jobs" in order to provide "SMAll Response Times" [34]. This class includes a range of practical variations other than those typically studied by theoreticians. The breadth of the class lies in the fact that it can be specified by 3 simple axiomatic properties that we describe below. To this point, SMART policies have been shown to have mean delay within a factor of two of optimal (SRPT) in the M/GI/1 setting [34] and are asymptotically equivalent with respect to delay distributions in the large buffer large deviations regime in the GI/GI/1 [19]. In this paper, we further characterize SMART policies by deriving the behavior of SMART policies in the many sources large deviations regime. In fact, we generalize the SMART classification and define the SMART-LD (SMART-Large Deviations) classification, which is obtained by dropping two of the three properties (the two "coherency" properties) in the SMART classification. We will show that, not only do all SMART policies have the same decay rate in the many sources regime, but, additionally, all SMART-LD policies also have the same decay rate.

This section is organized as follows. In Section 4.1, we define both the SMART and SMART-LD classes and contrast the SMART-LD class with the SMART class. Then, we discuss examples of policies included and excluded from the SMART-LD class in Section 4.2.

In Section 4.3, we present the basic intuition behind the main result for the SMART-LD class. Next, in Section 4.4, we present and prove our main result for SMART-LD policies (Theorem 1), which states that all SMART-LD policies have the same decay rate. Finally, we illustrate the behavior of the decay rate using numerical examples.

4.1 Defining the SMART-LD class

We will now formally describe the SMART-LD class. Denote jobs using a, b, and c where job a has original size s_a and remaining size r_a. SMART-LD is defined to be the set of scheduling policies that obey the following property.

Definition 1 All scheduling policies that satisfy the following property belong to the SMART-LD class.

(i) **Bias Property (LD):** *If $r_b \geq s_a$, then job a has priority over job b.*

Notice that the Bias Property guarantees that SMART-LD policies favor "small" jobs by guaranteeing that the job receiving service has smaller remaining size than the original size of all jobs in the system. This ensures that the server will not serve a new arrival with greater size than an existing one. This Bias Property ensures that SMART-LD policies do the "smart" thing.

It is important to contrast this definition with the definition of the SMART class, introduced in [34]. Formally, SMART is defined as follows.

Definition 2 All scheduling policies that satisfy the following properties belong to the SMART class.

(i) **Bias Property:** *If $r_b > s_a$, then job a has priority over job b.*
(ii) **Consistency Property:** *If job a ever receives service while job b is in the system, thereafter job a has priority over job b.*
(iii) **Transitivity Property:** *If an arriving job b preempts job c; thereafter, until job c receives service, every arrival, a, with size $s_a < s_b$ is given priority over job c.*

Comparing the definitions of SMART and SMART-LD, we notice a few things. First, notice that the Bias Property is slightly different in SMART and SMART-LD. The change to $r_b \geq s_a$ in the Bias Property (LD) guarantees a total ordering which is a key intuition behind the 2DQ framework. Second, notice that the Consistency and Transitivity Properties are not included in SMART-LD. Thus, SMART-LD is a significant extension to SMART, i.e., SMART-LD includes SMART.

The Consistency and Transitivity Properties in SMART essentially ensure the "coherency" of the priority scheme dictated by the Bias Property. In particular, the two properties are concerned with the priority scheme after jobs have received partial service. The Consistency Property effectively prevents time-sharing by enforcing the rule that once a job receives service, other jobs already in the system do not receive service until the former job leaves the system. In other words, if job a is given priority over job b, then job b will never served before job a. The Transitivity Property ensures that the SMART policies do not second guess the decision they have made. For example, if it is decided that job a is smaller (have higher priority) than job b, then new arrivals that are smaller than job a are considered smaller than job b.

The Consistency and Transitivity Properties are essential for proving that all SMART policies have mean delay within a factor of two of optimal and for the analysis of SMART policies in the large buffer regime. Thus, it is interesting that we can prove that the impact of the two properties that enforce "coherency" becomes insignificant when the scaling constant, N, scales to infinity in the many sources regime. The omission of the Consistency Property and the Transitivity Property give much more breadth to SMART-LD compared to SMART.

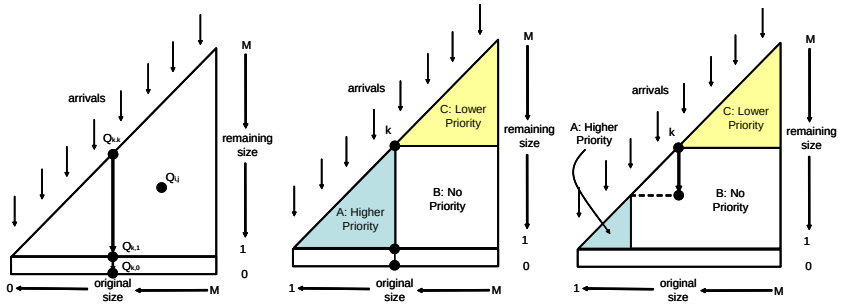
4.2 Policies included in SMART-LD

The SMART-LD class contains many important policies that "prioritize small jobs" such as SRPT, PSJF, and a wide array of hybrid policies with more complicated prioritization schemes. In particular, an interesting scheduling policy called RS policy, which assigns higher priority to jobs with smaller product of its remaining size and its original size, is included in SMART-LD. The RS policy is interesting in that the policy outperforms SRPT when we consider weighted mean delay measures such as the mean slowdown[3]. SMART-LD actually includes many generalizations of these policies as well. For example, it has been shown in [34] that scheduling policies that give priority based on a fixed priority function $p(s,r)$ such that for $s_1 \leq s_2$ and $r_1 < r_2$, $p(s_1,r_1) < p(s_2,r_2)$ are included in SMART-LD. An example of such a policy is a policy that has $p(s,r) = s^i r^j$ for all $i \geq 0$ and $j > 0$.

Apart from static priority policies, SMART-LD also includes time-varying policies, i.e., policies than can change their priority rules over time, based on system-state information, or randomization. These generalizations are possible because the SMART-LD definition enforces only a partial ordering on priorities of jobs in the system. It is of enormous practical importance that time-varying policies are included in SMART-LD, because it allows system designers to use the SMART-LD class in order to perform online multi-objective optimization. Specifically, suppose a system designer wants to optimize a secondary objective while still providing small delay. In order to accomplish this, the system designer can implement a parameterized version of SMART-LD, such as prioritizing based on $p(s,r) = s^i r^j$, and then use machine learning techniques to search the space (i,j) online for the SMART-LD policy that optimizes the secondary objective. (Note that i and j can be chosen to achieve SRPT, PSJF, and RS.) The elimination of the Consistency and Transitivity Properties by SMART-LD provides much greater freedom than SMART in the variation of the policy. In other words, SMART-LD contains policies that vary the priority after a job has been partially serviced, which is prohibited in SMART. For example, a policy in SMART-LD can allow a job that has been partially served to decide that it requires more/less service than its actual remaining processing time at the end of a time slot.

Despite its breadth, many policies that favor small jobs are excluded from SMART-LD, e.g., FB, and non-preemptive Shortest Job First (SJF). However, the exclusion of such policies is due to the goal in defining a class of policies that are near optimal in terms of delay across all service distributions and all loads. For example, SJF exhibits arbitrarily large delays when the second moment of the service distribution is large, and FB is the worst policy when the job size distribution is of increasing failure rate. One motivation for working with the class of SMART-LD policies is to illustrate the wide range of policies that behave like SRPT with respect to delay distribution in the many sources large deviations regime regardless of the job size distribution.

[3] Slowdown of a job is the delay divided by the job size.



(a) Job progression (b) Priority for incoming job (c) Priority for partial job

Fig. 3 Illustrations of the two dimensional queueing framework for SMART policies. The progression of a job in 2DQ from its arrival to the system until its departure is illustrated in (a). The priority structure for an incoming job is shown in (b) and for a partially served job is shown in (c).

4.3 SMART-LD and 2DQ framework

The SMART-LD class falls into the 2DQ framework very directly as follows. We arrange the virtual queues in a two dimensional grid where the x-axis is the original size of a job and the y-axis is the remaining size of a job (see Figure 3(a)). A new job arrives to queues in the uppermost diagonal strip, where original size and remaining size are equal, i.e., queues containing jobs that have not received any service. The job then progresses through the system by moving downward until the remaining size becomes 0 and leaves the system, i.e., the job has received service equal to its original size. We denote by $Q_{i,j}$, $i \geq j$, the queue that contains all jobs that were originally of size i and currently have remaining size j. Using this notation, a job of size k arrives at $Q_{k,k}$, and then moves through $Q_{k,k-1}$, $Q_{k,k-2}$, ..., $Q_{k,1}, Q_{k,0}$ as it receives service. We denote by $Q_{i,j}(t)$ the volume of jobs in $Q_{i,j}$ at time slot t. Additionally, we define Q_i as the queue containing all jobs of original size i, $Q_i = \bigcup_{j=1}^{i} Q_{i,j}$, and $Q_i(t) = \sum_{j=1}^{i} Q_{i,j}(t)$ similarly.

Let us now consider the behavior of a size k job under a SMART-LD policy. Upon arrival, the job resides in $Q_{k,k}$. By the Bias Property (LD), the following queues have higher and lower priority compared to $Q_{k,k}$:

$$\text{Higher Priority: } \bigcup_{i=1}^{k} \bigcup_{j=1}^{k-1} Q_{i,j}, \quad \text{Lower Priority: } \bigcup_{i=k+1}^{M} \bigcup_{j=k}^{M} Q_{i,j} \tag{4}$$

This is illustrated in Figure 3(b). We denote the group of higher priority queues as area A and lower priority queues as area C. Note that, there is an another area where the queues do not have any fixed priority order with respect to $Q_{k,k}$. Jobs in this area, B, can be serviced before or after the size k job. Due to the priority scheme, area A must be empty for a job in $Q_{k,k}$ to receive one unit of service. In other words, the delay experience by a job is dependent on the volume of jobs that are served before it, i.e., all jobs in area A and some jobs in area B. As depicted in Figure 3(c), when a job in $Q_{k,k}$ receives service it moves down the vertical line in the two dimensional queue and the priority areas change according to the position of the queue in which the job resides. For a partially serviced size k job to receive service, all queues in the corresponding area A, as depicted in Figure 3(c), must be empty. In both cases of an incoming size k job and partially serviced size k job, the relative

priority of queues in area B is unknown. However, the volume of jobs in area B can be sufficiently bounded to provide tight upper and lower bounds on the probability of delay in the many sources large deviations regime, yielding the delay distribution of SMART-LD.

4.4 The delay decay rate of SMART-LD

We now derive the actual delay decay rate of SMART-LD in the many sources regime. Our analysis of SMART-LD hinges on coupling the queueing dynamics of a SMART-LD policy to the queueing dynamics of a two-level priority queueing system through the 2DQ framework. A two-level priority queueing system consists of a pair of queues (high/low priority), where jobs in the low priority queue are served in a FCFS manner only if the high priority queue is empty. In such as system, jobs in the higher priority queue see only themselves and are served in a FCFS manner. Using the 2DQ framework, we show that a SMART-LD policy *partitions* the two dimensional collection of queues into three groups in any given time slot: a set with higher priority, a set with lower priority, and a set with unknown priority. This partition enables us to adapt the analysis of the two-level priority queueing system to analyze all SMART-LD policies. However, since directly connecting the analysis of SMART-LD to the two-level priority queueing system is difficult, we consider an intermediate system. For this purpose, we define a preemptive priority queueing system (denote as PRI) where there are M queues, with jobs of original size k arriving to queue-k. Queues corresponding to smaller jobs are granted higher priority over queues containing larger jobs. Further, jobs in PRI can not switch queues and each queue is served in FCFS order.

We first state the main theorem that describes the delay decay rate of SMART-LD. Let $W^{(N)}(k)$ denote the delay of size k job, and $I_{\overline{W}}(k,m)$ denote the actual delay decay rate of SMART-LD.

Theorem 1 *Let $\epsilon > 0$. For any $k \in \mathcal{M}$, the decay rate of delay for a size k job under any SMART-LD policy, $I_{\overline{W}}(k,m)$, satisfies*

$$I_{V_{C-\epsilon}}(k,m) \leq I_{\overline{W}}(k,m) \leq I_{V_C}(k,m), \tag{5}$$

where $I_{V_\mu}(k,m)$ is the virtual decay rate of delay under a priority queueing system, PRI, with capacity $N\mu$ and is defined as

$$I_{V_\mu}(k,m) = \inf_{T \geq 0} \left[\inf_{\mathbf{z}:\mathcal{Z}} \{\mathfrak{A}_{<k}(\mathbf{z}) + \mathfrak{A}_k\} \right], \tag{6}$$

where condition $\mathcal{Z}$ states that $\sum_{i=1}^{k} i z_i = \mu(T+m+1)$. Further,

$$\mathfrak{A}_{<k}(\mathbf{z}) = \sum_{i=1}^{k-1} I_{A_i}^{(-T,m)}(z_i)$$
$$\mathfrak{A}_k = I_{A_k}^{(-T,0)}(z_k).$$

This theorem states that asymptotically (in the large capacity and large number of flows regime), *all SMART-LD policies behave alike*, in that their delay decay rates are the same. In other words, for a job of original size k and for any fixed integer $m \geq 0$, we have that the delay distribution for $\overline{W}^{(N)}(k)$ is given by

$$P\left(\overline{W}^{(N)}(k) > m\right) = g(k,m)^N e^{-N I_{\overline{W}}(k,m)},$$

where $I_{\overline{W}}(k,m)$ *is the same for all SMART-LD policies.* Thus, the decay rate of any SMART-LD policy is the same as that of SRPT, which was derived in [36].

The decay rate in Equation (6) appears complicated, but does have intuition. It can be shown that in the many sources asymptote, the decay rate depends on the "most likely" way that the arrival processes deviate from their mean arrival rates in order to cause the delay exceeding m. Thus, the two infimums choose the most likely time scale (T) and partition of the overall arrival rate to job sizes ($\mathbf{z}$) for the event $\{W^{(N)}(k) > m\}$ to occur. Then, inside the infimums, $\mathfrak{A}_{<k}(\mathbf{z})$ and $\mathfrak{A}_k$ describe the amount of effect on the delay of a size k job by jobs arriving over the time interval $(-T, m)$ with size $< k$ and by jobs arriving over the time interval $(-T, 0)$ with size k. Note that jobs of size $> k$ do not affect the delay of size k jobs.

The proof of Theorem 1 requires tight upper and lower bounds of the decay rate which will be provided in the rest of this section. First, in Lemma 1, we derive upper and lower bounds of the delay decay rate of SMART-LD based on the virtual delay decay rates of SMART-LD. Based on this result, the proof of Theorem 1, which is presented after Lemma 1, provides the precise expressions of these bounds and shows that they are indeed tight. Let $I_{\overline{V}_\mu}(k,m)$ denote the decay rate of the virtual delay of a size k job under SMART-LD with total service rate $N\mu$. In addition, for bounding purposes, we consider the virtual delay of SMART-LD with capacity $N\mu$ where an additional size k job is inserted before the virtual job, and denote it as $I_{\tilde{V}_\mu}(k,m)$.

Lemma 1 *For any $k \in \mathcal{M}$, under any SMART-LD policy*

$$I_{\tilde{V}_C}(k,m) \leq I_{\overline{W}}(k,m) \leq I_{\overline{V}_C}(k,m). \tag{7}$$

Proof First, we derive the upper bound by showing

$$\Pr\left(\overline{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \leq \Pr\left(\overline{W}^{(N)}(k) > m\right). \tag{8}$$

Note that a virtual job (with size 0) need only to arrive at the front of the queue to be fully serviced. Thus, Equation (8) follows from the observation that, if a fictitious job did not leave the system (i.e. arrive at the head of the queue) before time m, then the actual job did not leave the queue. That is, the actual job did not even receive one unit of service. Thus, the actual job is guaranteed not to have left the system by time m. This queue state is depicted in Figure 4(a), where the last job corresponds to the actual job. Thus, we have

$$-\frac{1}{N}\log\Pr\left(\overline{W}^{(N)}(k) > m\right) \leq -\frac{1}{N}\log\Pr\left(\overline{V}^{(N)}(k) > m | A^N(0,0) > 0\right). \tag{9}$$

As $N \to \infty$, Equation (9) can be further upper bounded by $I_{\overline{V}_C}(k,m)$ using similar techniques to those in [30]. Finally,

$$-\frac{1}{N}\log\Pr\left(\overline{W}^{(N)}(k) > m\right) \underset{N\to\infty}{\longrightarrow} I_{\overline{W}}(k,m)$$

gives the upper bound.

To prove the lower bound, we add an extra size k job in front of the virtual queue and consider the virtual delay, i.e., $\tilde{V}^{(N)}(k)$. The queue state is depicted in Figure 4(b). We prove the inequality in (10) using a contra-positive argument.

$$\Pr(\overline{W}^{(N)}(k) > m) \leq \Pr\left(\tilde{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \tag{10}$$

The event $\left\{\tilde{V}^{(N)}(k) \leq m | A^N(0,0) > 0\right\}$ is equivalent to the statement that the virtual job leaves the system before time m. Note that for a virtual job to leave the system, all that is required is for the virtual job to reach the head of the queue. The virtual job reaches the head of the queue when the extra job of size k leaves $Q_{k,k}$, i.e., the extra job gets one unit of service. The key observation is that the extra job need not be fully served, only be partially served. However, for the extra job to be even partially serviced, the last job of the batch arrival must leave the system completely. This is due to the Bias Property which introduces the secondary ordering variable needed for the 2DQ framework: jobs with the same original size but with smaller remaining sizes have higher priority. This last job in the front of the virtual job is the job that represents the actual delay. Thus, we have the following:

$$\left\{\tilde{V}^{(N)}(k) \leq m | A^N(0,0) > 0\right\} \Rightarrow \left\{\overline{W}^{(N)}(k) \leq m\right\}$$

where $A \Rightarrow B$ denotes A implies B. This proves Equation (10) by a contra-positive argument. Thus we have

$$-\frac{1}{N}\log \Pr\left(\tilde{V}^{(N)}(k) > m | A^N(0,0) > 0\right) \leq -\frac{1}{N}\log \Pr\left(\overline{W}^{(N)}(k) > m\right).$$

As $N \to \infty$, above equation can be lower bounded by $I_{\tilde{V}_C}(k,m)$ using similar arguments to [30]. Finally, by noticing the definition of $I_{\overline{W}}(k,m)$, the proof is complete.

However, the precise expressions of $I_{\tilde{V}_C}(k,m)$ and $I_{\overline{V}_C}(k,m)$ are still unknown. Thus, in the following proof of Theorem 1, we derive the precise expressions of the above virtual delay decay rates and show that they are tight. Proving tight upper and lower bound of the decay rate is in many cases difficult without any additional assumptions. We make the following assumption on the arrival process:

Assumption 1 *We assume that the rate function of the arrival $A = \sum_{i=1}^{M} A_i$ satisfies*

$$I_A^{(-T,l)}\left(C(T+l+1)-v\right) < I_A^{(-T,0)}\left(C(T+1)\right), \tag{11}$$

for $v \in [v^* - \delta, v^* + \delta]$, $v^* > 0$ and $\delta > 0$ sufficiently small.

Assumption 1 is equivalent to the decay rate being additive [7]. Intuitively, a decay rate with the property of additive functionals implies that the occurrence of a rare event in the large deviation framework happens in a straight line. This assumption has been used extensively in large deviation literature [2, 15, 7].

We are now ready to prove Theorem 1. The proof is based on identifying a suitable bound on the potential service available to $Q_{k,k}$. For this purpose, we define $\overline{B}^N_{(k,k)}(a,b)$ as the volume of potential service that jobs in $Q_{k,k}$ can receive during the interval (a,b) under SMART-LD. Potential service corresponds to the maximum amount of service that can be received if the corresponding queue is never empty.

Proof (of Theorem 1) First, we derive the lower bound on the decay rate in Equation (5) by finding an upper bound of $\Pr\left(\tilde{V}^{(N)}(k) > m\right)$ and by using Lemma 1.

Let us consider the virtual delay in SMART-LD with the extra job, i.e., $\Pr\left(\tilde{V}^{(N)}(k) > m\right)$. Observe that if the virtual delay with the extra size k job exceeds m, then we have that the

(a) (b)

Fig. 4 Depiction of the queue state considered for the analysis of the (a) lower bound and (b) upper bound of SMART-LD.

queue length at time zero (i.e. $Q_{k,k}(0)$) and the extra job is not served by time m. In other words, $\left\{\tilde{V}^{(N)}(k) > m\right\} \Rightarrow \left\{Q_{k,k}(0) + k > \overline{B}^N_{(k,k)}(1,m)\right\}$, which implies

$$\Pr\left(\tilde{V}^{(N)}(k) > m\right) \leq \Pr\left(Q_{k,k}(0) + k > \overline{B}^N_{(k,k)}(1,m)\right). \tag{12}$$

From Loynes' formula, we have

$$\begin{aligned}\Pr\left(Q_{k,k}(0) + k > \overline{B}^N_{(k,k)}(1,m)\right) &= \Pr\left(\sup_{T\geq 0}\left[kA^N_k(-T,0) + k - \overline{B}^N_{(k,k)}(-T,m)\right] \geq 0\right) \\ &= \Pr\left(kA^N_k(-T^*_k,0) - \overline{B}^N_{(k,k)}(-T^*_k,m) + k \geq 0\right). \end{aligned} \tag{13}$$

Note that in addition to the volume of $Q_{k,k}(0)$, k is added in deriving Equation (13) due to the construction of $\tilde{V}$, i.e., addition of an extra size k job. Also, $-T^*_k$ is the most recent time in the past such that $Q_{k,k}(-T^*_k - 1) = 0$.

Now, we derive a lower bound on $\overline{B}^N_{(k,k)}(-T^*_k, m)$, which will in turn provides an upper bound of $\Pr(\tilde{V}^{(N)}(k) > m)$. We make use of the priority scheme of SMART-LD described through the 2DQ framework, which has been discussed in Subsection 4.3. It has been shown that area A is of higher priority compared to $Q_{k,k}$, and the queues in area B may or may not have higher priority. Thus a simple lower bound on $\overline{B}^N_{(k,k)}(-T^*_k, m)$ is the available service assuming that both areas A and B have higher priority. The volume of service that area A requires can be derived using the fact that all queues in area A are empty at time $-T^*_k - 1$, i.e., $Q_{(i,j)}(-T^*_k - 1) = 0$, for all $i \leq k$ and $j \leq k-1$. This follows immediately from Theorem 4.1 in [30], which is stated in the context of priority queues. Additionally, the volume of service that area B requires to be served before the tagged size k job receives full service and leaves the system is upper bounded by $\sum_{i=k+1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(-T^*_k - 1) + (T^*_k + m + 1)(M-1)$. This is due to the observation that all jobs in area B are partially served jobs. Note that at most a single partially serviced job can occur in a time slot, and the worst (largest) partially served job is of size $M-1$. Thus, after time $-T^*_k - 1$ at most $(T^*_k + m + 1)(M-1)$ can enter area B, which is exclusively populated by partially served jobs. Combining the two results, $\overline{B}^N_{(k,k)}(-T^*_k, m)$ is lower bounded as

$$\begin{aligned}\overline{B}^N_{(k,k)}(-T^*_k,m) \geq NC(T^*_k + m + 1) - \sum_{i=1}^{k}\sum_{j=1}^{k-1} Q_{(i,j)}(-T^*_k - 1) - \sum_{i=1}^{k-1} iA^N_i(-T^*_k, m) \\ - \sum_{i=k+1}^{M}\sum_{j=1}^{k-1} Q_{i,j}(-T^*_k - 1) - (T^*_k + m + 1)(M-1)\end{aligned}$$

$$= NC(T_k^* + m + 1) - \sum_{i=1}^{k-1} iA_i^N(-T_k^*, m) - \sum_{i=k+1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(-T_k^* - 1) - (T_k^* + m + 1)(M-1).$$

The quantity $\sum_{i=k+1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(-T_k^* - 1)$ is by definition the size of the queue $Q_{i,j}$ at time slot $-T_k^* - 1$, which is difficult to determine. However in the case of SMART, a simple bound exists for this quantity, i.e., $\sum_{i=k+1}^{M} \sum_{j=1}^{k-1} Q_{i,j}(-T_k^* - 1) \leq 1$. This is due to the fact that the Consistency and the Transitivity property in conjunction with the Bias property guarantees that the number of jobs larger than the tagged size k job possessing higher priority than the tagged job can be at most one at any time. This result was proven in Lemma 4.1 of [34]. However, the same argument does not hold when the Consistency and the Transitivity property are removed which is the case of SMART-LD. The partially served jobs in area B can have arbitrary priority amongst themselves. Consequently, the number of jobs in area B at time $-T_k^* - 1$ potentially having higher priority than the tagged size k job is no longer guaranteed to be at most one. Instead, the only possible guarantee that can be given for the number of jobs in area B that can affect the delay of the tagged job is simply all the jobs in $\sum_{i=k+1}^{M} \sum_{j=1}^{k} Q_{i,j}(-T_k^*)$. In particular, one can see that at time $-T_k^*$ at most $T_M^* - T_k^*$ jobs can be present in the *no priority* area, where $-T_M^*$ ($T_M^* \geq T_k^*$) is the last time before time 0 the *whole* system was empty. Thus, $\overline{B}_{(k,k)}^N(-T_k^*, m)$ can be bounded as

$$\overline{B}_{(k,k)}^N(-T_k^*, m) \leq NC(T_k^* + m + 1) - \sum_{i=1}^{k-1} iA_i^N(-T_k^*, m) - (T_M^* - T_k^*)(M-1) - (T_k^* + m + 1)(M-1). \quad (14)$$

To save space, we denote $\mathcal{T}(a, b)$ as the event $\{kA_k^N(-a, 0) + \sum_{i=1}^{k-1} iA_i^N(-a, m) - NC(a+m+1)+(b+m+1)M > 0\}$. From Equation (12), Equation (13), and Equation (14), we have

$$\begin{aligned}
\Pr\left(\tilde{V}^{(N)}(k) > m\right) &\leq \Pr\left(kA_k^N(-T_k^*, 0) + k - \overline{B}_{(k,k)}^N(-T_k^*, m) > 0\right) \\
&\leq \Pr\Bigg(kA_k^N(-T_k^*, 0) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*, m) - NC(T_k^* + m + 1) \\
&\qquad (T_M^* - T_k^*)(M-1) + (T_k^* + m + 1)(M-1) + k > 0\Bigg) \\
&\leq \Pr\Bigg(kA_k^N(-T_k^*, 0) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*, m) - NC(T_k^* + m + 1) \\
&\qquad +(T_M^* + m + 1)M > 0\Bigg) \\
&= \Pr\left(\mathcal{T}(T_k^*, T_M^*)\right) \\
&\leq \Pr\left(\bigcup_{T \geq 0} \mathcal{T}(T_k^*, T)\right) \\
&\leq \sum_{T=0}^{\infty} \Pr\left(\mathcal{T}(T_k^*, T)\right) \\
&= \sum_{T=0}^{\infty} \Pr\left(\mathcal{T}(T_k^*, T_M^*) \cap \{T_M^* = T\}\right) \\
&= \sum_{T=0}^{L} \Pr\left(\mathcal{T}(T_k^*, T_M^*) \cap \{T_M^* = T\}\right)
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{T=L+1}^{\infty} \Pr\left(\mathcal{T}(T_k^*, T_M^*) \cap \{T_M^* = T\}\right) \\
&\leq \sum_{T=0}^{L} \Pr\left(\mathcal{T}(T_k^*, l)\right) + \sum_{T=L+1}^{\infty} \Pr\left(T_M^* = T\right) \\
&\leq L \Pr\left(\mathcal{T}(T_k^*, L)\right) + 2g(k,m)^N e^{-NLI_A^{(0,0)}(C)}, \qquad (15)
\end{aligned}
$$

for any finite L and for some $g(k,m)^N$ which satisfies $\lim_{N\to\infty} -\frac{1}{N}\log g(k,m)^N = 0$. The last inequality in Equation (15) is the result of the following two arguments. The first term, $\Pr\left(\mathcal{T}(T_k^*, l)\right)$, is an increasing function with respect to l. The upper bound on the second term is based on the observation that $P(T_M^* = T) \leq P\left(A^N(-T,0) \geq NC(T+1)\right)$ and Assumption 1. In particular, we have

$$
\begin{aligned}
P(T_M^* = T) &\leq P\left(A^N(-T,0) \geq NC(T+1)\right) \\
&\leq g(k,m)^N e^{-NI_A^{(-T,0)}(C(T+1))} \\
&\leq g(k,m)^N e^{-N(T+1)I_A^{(0,0)}(C)},
\end{aligned}
$$

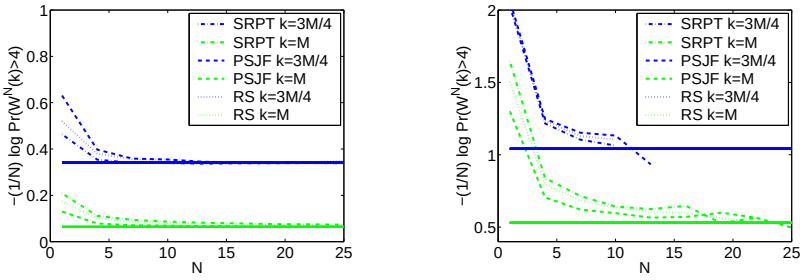
and thus

$$
\begin{aligned}
\sum_{T=L+1}^{\infty} \Pr\left(T_M^* = T\right) &\leq \sum_{T=L+1}^{\infty} g(k,m)^N e^{-N(T+1)I_A^{(0,0)}(C)} \\
&\leq 2g(k,m)^N e^{-NLI_A^{(0,0)}(C)}.
\end{aligned}
$$

Finally, to complete the lower bound part of the proof, we show that the first term in Equation (15) has a rate function that can be lower bounded by $I_{V_{C-\epsilon}}(k,m)$, and that the second term has a smaller rate function than $I_{V_{C-\epsilon}}(k,m)$ and thus can be effectively ignored. First, we show that the rate function of the first term in Equation (15) is lower bounded by $I_{V_{C-\epsilon}}(k,m)$. Fixing any $\epsilon, L > 0$, it can be shown that for $N > \frac{L+m+1}{\epsilon(T_k^*+m+1)}$,

$$
\begin{aligned}
\Pr\left(\mathcal{T}(T_k^*, L)\right) &= \Pr\Bigg(kA_k^N(-T_k^*,0) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*,m) - NC(T_k^*+m+1) \\
&\qquad +(L+m+1)M > 0\Bigg) \\
&= \Pr\Bigg(kA_k^N(-T_k^*,0) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*,m) \\
&\qquad -N\left(C - \frac{L+m+1}{N(T_k^*+m+1)}\right)(T_k^*+m+1) > 0\Bigg) \\
&\leq \Pr\Bigg(kA_k^N(-T_k^*,0) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*,m) \\
&\qquad -N(C-\epsilon)(T_k^*+m+1) > 0\Big). \qquad (16)
\end{aligned}
$$

Note that Equation (16) is the expression for the decay rate of size k jobs in PRI having capacity $C-\epsilon$. Using similar techniques as in [6,12], it follows that the lower bound of the decay rate of the first term in Equation 15, $\Pr\left(\mathcal{T}(T_k^*, L)\right)$, is $I_{\overline{V}_{C-\epsilon}}(k,m)$. Lastly, since the above result holds for any $L > 0$, we can select L such that it satisfies $L \geq \frac{I_{C-\epsilon}}{I_A^{(0,0)}(C)}$ so that $e^{-NLI_A^{(0,0)}(C)} \leq e^{-NI_{\overline{V}_{C-\epsilon}}(k,m)}$ and the lower bound of the decay rate of $\Pr(\tilde{V}^{(N)}(k) >$



(a) High load (b) Low load

Fig. 5 Plot of the rate of convergence of the decay rate of SMART-LD under the uniform workload with $M = 16$, and $m = 4$ when the load is high ($\rho = 0.8$) and when load is low ($\rho = 0.2$). The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation, since a large delay for smaller job sizes is a very low probability event as N grows.

m) is $I_{\overline{V}_{C-\epsilon}}(k,m)$ through the contraction principle. Applying Lemma 1, we have the lower bound on the decay rate in Equation (5).

Next, we derive the upper bound on the decay rate in Equation (5) by deriving a lower bound on $\Pr(V^{(N)}(k) > m)$ and combining it with the result of Lemma 1. We do so by comparing the SMART-LD class with a priority queueing system which lower bounds the delay experienced by the job.

Consider a PRI system with capacity NC, which we describe again. This system consists of M queues, with jobs of original size k arriving to queue-k. Higher priority is given to queues corresponding to smaller jobs. Partially served jobs in this system continue to reside in the same queue and jobs in each queue are served in a FCFS manner. In comparing the PRI system to SMART-LD, we can think of PRI as a SMART-LD scheme where $Q_k = \sum_{i=1}^{k} Q_{(k,i)}$ and priorities are assigned such that area A has higher priority whereas area B and C have lower priority.

Denote $V^{(N)}(k)$ as the virtual delay of a size k job for PRI. Then, the event $\{V^{(N)}(k) \leq m\}$ of PRI ensures the event $\{\overline{V}^{(N)}(k) \leq m\}$ of SMART-LD policies. This comes from the fact that the external arrival to both PRI and SMART-LD are the same but the residual service available to $Q_{k,k}$ in SMART-LD is upper bounded by the residual service for queue-k of PRI. This follows from the fact that the residual service of SMART-LD is the remaining service after servicing of all of area A and possibly a part or all of area B. However, the residual service in PRI is that of after servicing only the area A, and none of area B. Thus, PRI provides a lower bound on the virtual delay of a job compared to SMART-LD. In other words, we have $I_{\overline{V}_C}(k,m) \leq I_V(k,m)$ and combining it with Lemma 1, the proof is complete.

4.5 Numerical results

The resulting rate function for SMART-LD described in Theorem 1 is difficult to understand due to its complicated form. In this subsection, we attempt to increase understanding of the delay decay rate $I_{\overline{W}}(k,m)$, and thus $Pr(\overline{W}(k) > m)$, of SMART-LD by illustrating its numerical values when the load (ρ), the variability of the service distribution, the range of the service distribution (M), and the threshold value (m) are varied.

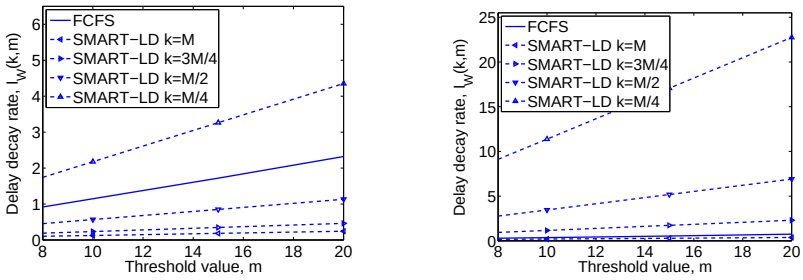


Fig. 6 Plot of the decay rate as a function of the threshold m under the (a) power-law and (b) high variability workload under both SMART-LD and FCFS with the maximum job size $M = 16$ and high load ($\rho = 0.8$). Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. Note that since decay rates of size 1 jobs are infinite, they are omitted.

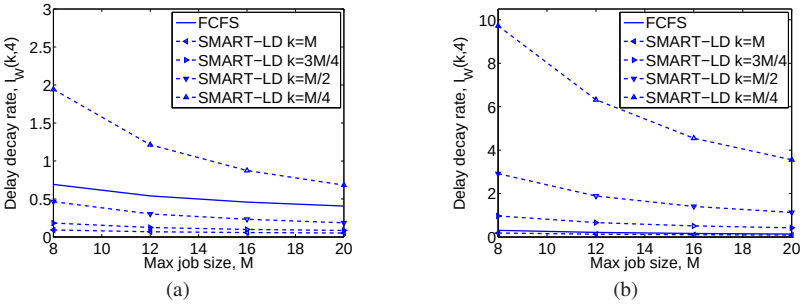
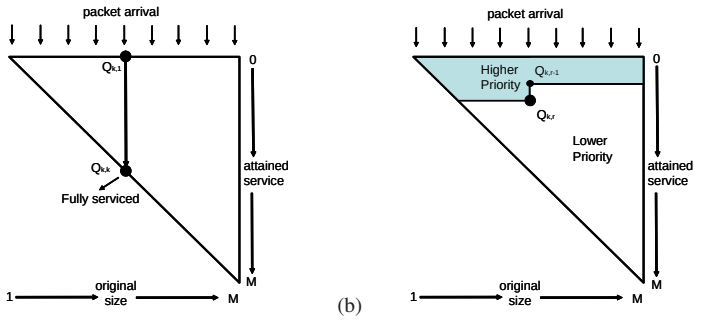


Fig. 7 Plot of the decay rate as a function of the maximum job size M under the (a) power-law and (b) high variability workload under both SMART-LD and FCFS with the threshold $m = 4$ and high load ($\rho = 0.8$). Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. Note that since decay rates of size 1 jobs are infinite, they are omitted.

The setup of the experiment is as follows. We assume that there are jobs of sizes 1, $M/4$, $M/2$, $3M/4$, and M. Job arrivals are on-off processes in which a job of size k arrives in a time-slot with probability p_k and does not with probability $1 - p_k$. The distribution of job sizes is controlled through p_k, where we consider *uniform*, *power-law*, and *high variability* distributions. The *uniform* distribution denotes the case where each job size is equally likely. In the *power-law* case, the job size distribution follows the power-law distribution with exponent 2, i.e., a discrete and truncated counterpart of the Pareto distribution (a well known heavy-tailed distribution). Due to the small spread of job sizes, the power-law distribution is not highly variable; thus, to study the impact of high variability, we also consider a distribution where the largest job make up half the load (analogous to the observation made in web servers where the largest 1% of jobs make up half the load). Thus, we consider the *high variability* workload where size 1, $M/4$, $M/2$, $3M/4$, and M job arrivals make up $1/24$, $1/12$, $1/8$, $1/4$, and $1/2$ of the total load. Lastly, we assume that the per source capacity is 1 and consider load less than 1 to ensure stability.

Figure 5 illustrates the convergence of the delay distribution as the scaling constant, N, increases under three common policies in the SMART-LD class. We specifically consider the uniform distribution. The solid lines are the numerical calculations of the delay decay



(a) Job progression (b) (b) Priority scheme

Fig. 8 Illustrations of the two dimensional queueing framework for FB. The progression of a job between queues while it is in the system is illustrated in (a). The priority structure for a job in $Q_{k,r}$ is shown in (b).

rate described in Theorem 1. The other dotted lines are generated using an event-driven simulation for a finite number of flows, i.e., N. The simulation matches the uniform workload described above except that a Poisson arrival process is used. As can be seen in Figure 5, the empirical results approach that of the asymptotic limit for $N = 20$. Thus we can conclude that the *derived asymptotic delay decay rate is accurate and thus useful for a realistic setting* where web-servers and routers are typically accessed by much more than 20 simultaneous flows.

In Figures 6 and 7, we investigate the behavior of the decay rate of SMART-LD as m and M are varied, respectively, for high load ($\rho = 0.8$). Note that we have also investigated other loads, to verify that $\rho = 0.8$ is a representative choice. These plots illustrate the effect of increasing m and M on the decay rate of delay. In particular, as the threshold value m increases, Figure 6 shows that the decay rate increases, and thus $Pr(W(k) > m)$ decreases. As the maximum job size M increases, Figure 7 shows that the decay rate of all job sizes decreases, which is not surprising since this leads to an increase in service times for all job sizes.

The final observation that we make is that, in each of the plots, small job sizes have much better decay rates under SMART-LD policies than under FCFS; whereas large job sizes have better decay rates under FCFS than under SMART-LD. Figure 6 illustrates that the job sizes at which SMART-LD becomes better or worse than FCFS are highly dependent on the service distribution and load. This behavior of SMART-LD will be further discussed in Section 6.2.

5 Foreground-Background scheduling

There are many cases where designers want to give priority to small jobs, but do not have any information about the job sizes. In these settings, Foreground-Background (FB) serves as a "poor man's SRPT," and many cases, for example in routers and operating systems, FB has led to significant improvements in delay performance [22,23,8].

FB is a preemptive scheduling policy that serves jobs with the smallest attained service (age) first. When there are multiple jobs with the same attained service, the capacity of the server is shared. In the discrete time model discussed in this paper, FB shares the capacity amongst the jobs with least attained service by giving each job one unit of service (the

smallest amount of service available). Note that a newly arriving job always preempts the job (or jobs) currently in service and retains the processor until one of the following occurs: (i) the job departs, (ii) the next arrival appears, or (iii) the job has obtained an amount of service equal to that received by the job(s) preempted on its arrival.

Thus, under FB, small jobs can be completely served without queueing behind large jobs. This heuristic is effective because computer workloads often have a decreasing failure rate, from which it follows that jobs that have a large attained service are likely to have a large remaining size as well. Thus, by prioritizing jobs with the least attained service, FB is prioritizing jobs that are likely to have smaller remaining sizes. In fact, it has been proven that FB minimizes mean queue length and mean delay among blind policies [25,26].

The analysis of FB has a long history, and we refer the reader to the recent survey by Nuyens & Wierman [18] for more details. For our purposes, it is enough to note that FB has been analyzed in the large buffer regime [3,14,16,17], but not in the many sources regime.

The remainder of this section is organized as follows. We will first describe how FB fits into the 2DQ analysis framework (Section 5.1). Then, we will present our analysis of the decay rate of FB (Section 5.2). Finally, we will illustrated the behavior of the decay rate using numeric experiments (Section 5.3).

5.1 FB and 2DQ framework

In a similar manner to the SMART-LD case, we add original job size as a secondary variable. This means that, in the event of ties (having the same attained service), instead of sharing the server among the jobs with equal attained service, the job with the smallest original size is served a unit first. Further, jobs that have the same original size and the same attained service are served according to a FCFS rule. Keep in mind that jobs are not served with the full service capacity, but are only serviced a single unit at once, which is consistent with the discrete version of PS. Note that using the job size as a secondary ordering variable does not alter the performance of FB in the asymptotic framework, it is simply a modeling decision used to make the analysis tractable.

Formally, a queue $Q_{i,j},\ i \geq j$, denotes the queue that contains all the jobs having original size i that have received j unit of service. Thus a job of original size k first arrives to $Q_{k,0}$ and then progresses to $Q_{k,1}$, $Q_{k,2}$ ... $Q_{k,k-1}, Q_{k,k}$, at which point the job is fully serviced and leaves the system. This is depicted in Figure 8. We again denote $Q_{i,j}(t)$ as the volume of $Q_{i,j}$ at time t, and $Q_i(t)$ as the volume of the queue that contains all jobs that have original size i, Q_i, where $Q_i = \bigcup_{j=0}^{j=i-1} Q_{i,j}$.

Let us now consider a tagged size k job that arrives to the system at time 0 and has received r units of service (see Figure 8). By the definition of FB, all jobs that have been served less than r units have higher priority than the tagged job. Additionally, jobs with smaller original size which have the same attained service r also have higher priority. In other words, for any job in $Q_{k,r}$ to be serviced an additional unit, the following queues must be empty:

$$\bigcup_{i=1}^{k-1} \bigcup_{j=0}^{r} Q_{i,j} + \bigcup_{i=k+1}^{M} \bigcup_{j=0}^{r-1} Q_{i,j} + \bigcup_{j=0}^{r-1} Q_{k,j}$$

Further, since in each queue, $Q_{i,j}$, jobs are serviced in FCFS order, jobs that arrived to Q_k before the tagged job must be serviced r units, and jobs that arrived after the tagged job must be served only $r-1$ units. The same argument can be made for $Q_{k,k-1}$, which is the queue for size k jobs that will leave the system once it is served again a unit.

5.2 The delay decay rate of FB

In this section, we present the delay decay rate of FB and provide the proof. The delay decay rate of FB in the many sources large deviations regime is described in Theorem 2, and the proof is provided in the rest of this section. We denote $I_{\hat{W}}(k,m)$ as the delay decay rate of size k jobs under the FB scheduling policy.

Theorem 2 *Under Assumptions 2 and 3, the decay rate of delay for size k jobs under FB is*

$$I_{\hat{W}}(k,m) = \inf_{T\geq 0}\left[\inf_{\mathbf{y}:\mathcal{Y}}\left(\mathfrak{A}_{<k}(\mathbf{y}) + \mathfrak{A}_k(\mathbf{y}) + \mathfrak{A}_{>k}(\mathbf{y})\right)\right], \tag{17}$$

where condition $\mathcal{Y}$ states that $\sum_{i\in\hat{\mathbf{k}}} iy_i + ky_k + \sum_{i\in\check{\mathbf{k}}}(k-1)y_i = C(T+m+1)$ with $\hat{\mathbf{k}}\{1,\ldots,k-1\}$, $\check{\mathbf{k}} = \{k+1,\ldots,M\}$, and $y_k^{(1)} + \frac{k-1}{k}y_k^{(2)} = y_k$. Further,

$$\begin{aligned}
\mathfrak{A}_{<k}(\mathbf{y}) &= \sum_{i=1}^{k-1} I_{A_i}^{(-T,m)}(y_i)\\
\mathfrak{A}_k(\mathbf{x}) &= I_{A_k}^{(-T,0)}(y_k^{(1)}) + I_{A_k}^{(1,m)}(y_k^{(2)})\\
\mathfrak{A}_{>k}(\mathbf{y}) &= \sum_{j=k+1}^{M} I_{A_i}^{(-T,m)}(y_j).
\end{aligned}$$

Theorem 2 characterizes the asymptotic delay distribution of FB in the many sources regime. Though the form of Equation (17) is complicated, we can obtain intuition for it. Again, the decay rate depends on the "most likely" way that the arrival processes deviate from their mean arrival rates in order to cause the delay to exceed m. Thus, the two infimums choose the most likely time scale (T) and arrival rates for each job size ($\mathbf{y}$), where y_k is separated into the arrivals before ($y_k^{(1)}$) and after ($y_k^{(2)}$) the tagged arrival. Then, inside the infimums, $\mathfrak{A}_{<k}(\mathbf{y})$, $\mathfrak{A}_k(\mathbf{y})$, and $\mathfrak{A}_{>k}(\mathbf{y})$ characterize the contribution to the delay of a size k job made by jobs with size $< k$, jobs of size k, and jobs of size $> k$ arriving in the time interval $(-T,m)$. This intuition points out one key difference between the decay rates of $W(k)$ under SMART-LD and FB. While under FB $\mathfrak{A}_{>k}(\mathbf{y})$ characterizes the effect of jobs with size larger than k, there is no such term in the decay rate of SMART-LD (see Equation (6)).

In the rest of the section, we provide a formal proof of Theorem 2 using the 2DQ framework. We start the analysis of FB by deriving an upper bound on the probability of virtual delay under FB, i.e., lower bound of the delay decay rate. Denote the virtual delay of size k job under FB as $\hat{V}^{(N)}(k)$. We bound the probability that the virtual delay for size k jobs exceeds m, $\Pr(\hat{V}^{(N)}(k) > m)$, as follows.

Lemma 2 *For any $k \in \mathcal{M}$, the virtual delay of size k jobs in FB satisfies*

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right)\\
&= \Pr\left(kA_k^N(-T_k^*,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T_k^*,m) > 0\right), \qquad (18)
\end{aligned}$$

where $A_k^N(-T,0)$ is the number of size k job arrivals in the interval $(-T,0)$, $\hat{B}_k(-T,m)$ is the service available to Q_k during $(-T,m)$, and T_k^* is the last time before 0 that satisfies $Q_k(-T_k^*-1) = 0$.

The lemma states that a *possible* scenario in which the event $\{\hat{V}^{(N)}(k) > m\}$ could occur is when the total volume of arrivals for size k job before the virtual job arrives and a portion ($\frac{k-1}{k}$) of the volume of arrivals of size k jobs after the arrival of the virtual job, exceed the available capacity for Q_k.

Proof Consider Q_k. By the operation of FB, $\{\hat{V}^{(N)}(k) > m\}$ implies that all jobs in $Q_k(0)$ have not been fully serviced by time m, i.e., $\{kA_k^N(-T_k^*, 0) > \hat{B}_k^N(-T_k^*, m)\}$. Since adding more jobs makes the event of exceeding the available capacity more probable, we have

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) - hatB_k^N(-T,0)\right) - \hat{B}_k^N(1,m) > 0\right) \\
&= \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) - \hat{B}_k^N(-T,m)\right) > 0\right) \\
&\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right).
\end{aligned}$$

Finally, Equation (18) follows from Loynes' formula.

Note that the total available capacity to Q_k in interval $(-T, m)$ with regards to the event $\{\hat{V}^{(N)}(k) > m\}$, i.e. $\hat{B}_k^N(-T, m)$, is the remaining capacity after all the higher priority queues are served in FB. So, the following holds:

$$\begin{aligned}
\hat{B}_k^N(-T,m) \geq NC(T+m+1) &- \sum_{i=1}^{k-1} iA_i^N(-T,m) - \sum_{i=1}^{k-1} Q_i(-T-1) \\
&- \sum_{i=k+1}^{M}(k-1)A_i^N(-T,m) - \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(-T-1). \qquad (19)
\end{aligned}$$

Using Lemma 2 and the above bound on $\hat{B}_k^N(-T, m)$, we derive the following theorem.

Theorem 3 *The virtual decay rate of size k jobs under FB is bounded as*

$$I_{\hat{V}}(k,m) \geq \inf_{T\geq 0}\left[\inf_{\mathbf{y}:\mathcal{Y}}\left\{\inf_{\mathbf{x}:\mathcal{X}}\left(\mathfrak{A}_{<k}(\mathbf{y}) + \mathfrak{A}_k(\mathbf{y}) + \mathfrak{A}_{>k}(\mathbf{y})\right)\right\}\right], \qquad (20)$$

where $\mathcal{Y}$, $\mathcal{X}$, $\mathfrak{A}_{<k}(\mathbf{y})$, $\mathfrak{A}_k(\mathbf{y})$, and $\mathfrak{A}_{>k}(\mathbf{y})$ are defined in Theorem 2.

Proof Combining Equation (19) with Lemma 2, we have the following upper bound on the probability of the event $\{\hat{V}^{(N)}(k) > m\}$:

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\leq \Pr\left(\sup_{T\geq 0}\left(kA_k^N(-T,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T,m)\right) > 0\right) \\
&= \Pr\left(kA_k^N(-T_k^*,0) + (k-1)A_k^N(1,m) - \hat{B}_k^N(-T_k^*,m) > 0\right) \\
&\leq \Pr\Bigg(kA_k^N(-T_k^*,0) + (k-1)A_k^N(1,m) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*,m) \\
&\qquad + \sum_{i=1}^{k-1} Q_i(-T_k^*-1) + \sum_{i=k+1}^{M}(k-1)A_i^N(-T_k^*,m)
\end{aligned}$$

$$
\begin{aligned}
&\quad + \sum_{i=k+1}^{M} \sum_{j=0}^{k-2} Q_{i,j}(-T_k^* - 1) - NC(T_k^* + m + 1) > 0 \Bigg) \\
&= \Pr\Bigg(kA_k^N(-T_k^*, 0) + (k-1)A_k^N(1, m) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*, m) \\
&\quad + \sum_{i=k+1}^{M} (k-1)A_i^N(-T_k^*, m) - NC(T_k^* + m + 1) > 0 \Bigg) \qquad (21) \\
&\leq \Pr\Bigg(\bigcup_{T \geq 0} \Bigg(kA_k^N(-T, 0) + (k-1)A_k^N(1, m) + \sum_{i=1}^{k-1} iA_i^N(-T, m) \\
&\quad + \sum_{i=k+1}^{M} (k-1)A_i^N(-T, m) - NC(T + m + 1) \Bigg) > 0 \Bigg) \\
&\leq \sum_{T \geq 0} \Pr\Bigg(kA_k^N(-T, 0) + (k-1)A_k^N(1, m) + \sum_{i=1}^{k-1} iA_i^N(-T, m) \\
&\quad + \sum_{i=k+1}^{M} (k-1)A_i^N(-T, m) - NC(T + m + 1) > 0 \Bigg),
\end{aligned}
$$

where Equation (21) follows from the observation that $Q_i(-T_k^* - 1) = 0$ for $1 \leq i \leq k-1$ and $Q_{i,j}(-T_k^* - 1) = 0$ for $k+1 \leq i \leq M$, $0 \leq j \leq k-2$. The justification is similar to that of a priority queueing system. Namely, since the above queues have higher priority than Q_k, when Q_k is empty all higher priority queues must be empty. Applying the contraction principle completes the proof.

Theorem 3 provides a *possible* scenario in which $\{\hat{V}^{(N)}(k) > m\}$ could occur. To show that this scenario is indeed the most dominant and controls the behavior of the probability, we will show that the same scenario provides the upper bound on the decay rate of $\{\hat{V}^{(N)}(k) > m\}$, i.e., lower bound of the tail probability.

Thus, we now develop an upper bound of the decay rate of FB which is tight with the lower bound derived in Theorem 3, i.e., we develop the tight lower bound for the tail probability. The main argument for the upper bound is that, by using the 2DQ framework, FB can be viewed as a complex but tractable time varying priority queueing system, as was in the case with SMART-LD. Thus, using similar analysis, we derive a tight upper bound for the decay rate.

Theorem 4 *The probability of the virtual delay for size k jobs in FB can be lower bounded as*

$$
\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) \geq \Pr\Bigg(&\inf_{0 \leq l \leq m} \Bigg\{ \sum_{i=1}^{k-1} iA_i^N(-T_k^*, l) + kA_k^N(-T_k^*, 0) + (k-1)A_k^N(1, l) \\
&+ \sum_{i=k+1}^{M} (k-1)A_i^N(-T_k^*, l) - NC(T_k^* + l + 1) \Bigg\} > 0 \Bigg), \qquad (22)
\end{aligned}
$$

where $-T_k^*$ is the last time before time 0 when $Q_k(-T_k^* - 1) = 0$.

Proof As explained in the previous subsection, there exists a group of queues and arrivals that constitute higher priority compared to the virtual job that arrives at time 0. In particular, the volume of jobs in higher priority queues and arrivals with respect to the virtual job at time l is

$$\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(l) + \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(l) + \sum_{j=0}^{k-2} Q_{k,j}(l) + A_k(-T_k^*, 0). \tag{23}$$

If the higher priority queues and arrivals with respect to the virtual job expressed in Equation (23), are never empty at any time during $(0, m)$, then the virtual job is guaranteed *not* to leave the system in the interval $(0, m)$. Based on this observation we derive a lower bound on $\Pr(\hat{V}^{(N)}(k) > m)$ as follows:

$$\begin{aligned}
\Pr\left(\hat{V}^{(N)}(k) > m\right) &\geq \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(l) + \sum_{i=k+1}^{M}\sum_{j=0}^{k-2} Q_{i,j}(l) \right.\right. \\
&\qquad \left.\left. + \sum_{j=0}^{k-1} Q_{k,j}(l) + A_k(-T_k^*, 0)\right\} > 0\right) \\
&= \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} Q_{i,j}(-T_k^*-1) + \sum_{i=1}^{k-1} iA_i^N(-T_k^*, l)\right.\right. \\
&\quad + \sum_{i=k+1}^{M}\sum_{j=0}^{k-1} Q_{i,j}(-T_k^*-1) + \sum_{i=k+1}^{M}(k-1)A_i^N(-T_k^*, l) \\
&\quad \left.\left. + kA_k^N(-T_k^*, 0) + (k-1)A_k^N(1, l) - NC(T_k^* + l + 1)\right\} > 0\right) \\
&= \Pr\left(\inf_{0\leq l\leq m}\left\{\sum_{i=1}^{k-1} iA_i^N(-T_k^*, l) + \sum_{i=k+1}^{M}(k-1)A_i^N(-T_k^*, l)\right.\right. \\
&\quad \left.\left. + kA_k^N(-T_k^*, 0) + (k-1)A_k^N(1, l) - NC(T_k^* + l + 1)\right\} > 0\right).
\end{aligned}$$

In the calculation above, note that we make use of the previous argument that at time $-T_k^* - 1$ all higher priority queues are empty.

Extending the above result to obtain a tight lower bound on the decay rate of FB is difficult without further assumptions on the inputs. We make two assumptions in order to complete the proof of Theorem 2.

Assumption 2 *Let A_{high} denote the sum of all higher priority arrivals described in Theorem 4. Then we assume that the corresponding rate function satisfies*

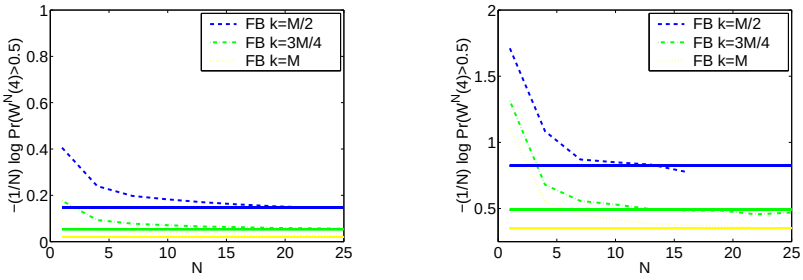
$$I_{A_{high}}^{(-T_k^*, l)}\left(C(T_k^* + l + 1) - v\right) < I_{A_{high}}^{(-T_k^*, 0)}\left(C(T_k^* + 1)\right), \tag{24}$$

for $v \in [v^* - \delta, v^* + \delta]$, $v^* > 0$, and $\delta > 0$ sufficiently small.

Assumption 3 *Define*

$$\mathbf{A}_{high} = \left(\ldots, A_{high}^N(-T, 0), \ldots, A_{high}^N(-1, 0), A_{high}^N(0, 0)\right).$$

Then, we assume that the stochastic process $\mathbf{A}_{high}$ satisfies



(a) High load (b) Low load

Fig. 9 Plot of the rate of convergence to the decay rate of FB under the uniform workload with $M = 16$, and $m = 4$ when the load is high ($\rho = 0.8$) and when the load is low ($\rho = 0.2$). The asymptotic decay rate is shown as a dotted line. Note that only the decay rates of the larger sizes are shown because only these can be estimated accurately in simulation, since a large delay for smaller job sizes is a very low probability event as N grows.

$$\left(\mathbf{A}_{high}|A^N_{high}(0,0) = 0\right) \leq_{st} \left(\mathbf{A}_{high}|A^N_{high}(0,0) > 0\right),$$

where $\mathbf{A}_{high}$ *was defined in Assumption 2.*

Assumption 2 is similar to Assumption 1 in Section 4. However, Assumption 2 describes the additive property for a subset of arrivals which represents all higher priority arrivals with respect to the tagged size k job under the FB scheduling policy.

Assumption 3 allows us to show that the virtual delay decay rate is equal to the actual delay decay rate under FB. This assumption essentially says that the arrival process has the property that if there are very few arrivals in a given time slot, there were also very few arrivals in the immediate past (and vice versa). It is a kind of "burstiness" assumption for the input arrival.

We are now ready to complete the proof of Theorem 2.

Proof (of Theorem 2) It follows from Theorem 4 and Assumption 2 that $I_{\hat{V}}(k,m)$ is upper bounded by the expression in Equation (17) using the same technique as in [7]. Further, applying Theorem 3, we obtain equality. Lastly, using similar arguments as in [30], we can conclude that the actual delay decay rate is equal to the virtual delay decay rate under Assumption 3, which completes the proof.

5.3 Numerical results

Similar to the SMART-LD section, we attempt to increase understanding of the derived rate function of FB as described in Theorem 2 by evaluating the numerical values when the load (ρ), the variability of the service distribution, the range of the service distribution (M), and the threshold value (m) are changed under the same setup.

Figure 9 illustrates the convergence of the delay distribution as the scaling constant, N, increases. As can be seen in Figure 9, the empirical results approach the theoretical asymptotic limit derived in Theorems 2 for $N = 20$. Thus, again we can conclude that

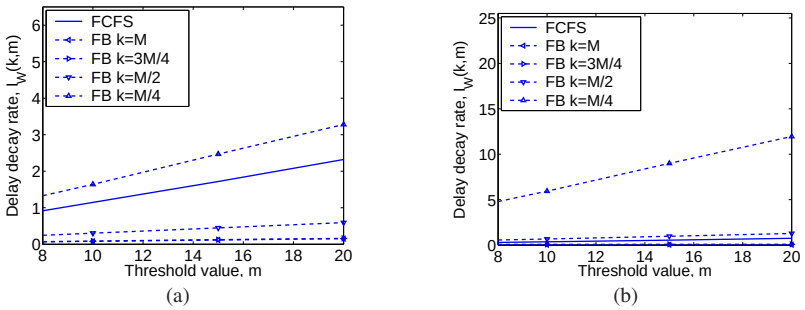


Fig. 10 Plot of the decay rate as a function of the threshold m under the (a) power-law and (b) high variability workload under both FB and FCFS with the maximum job size $M = 16$ and high load ($\rho = 0.8$). Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. Note that since decay rates of size 1 jobs are infinite, they are omitted.

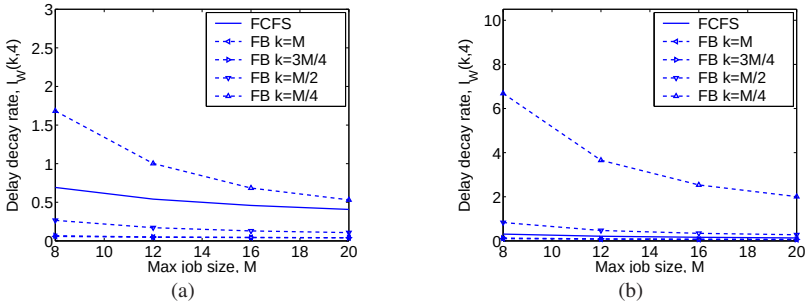


Fig. 11 Plot of the decay rate as a function of the maximum job size M under the (a) power-law and (b) high variability workload under both FB and FCFS with the threshold $m = 4$ and high load ($\rho = 0.8$). Each line in the figures corresponds to the decay rate of delay experienced by a specific job size k. Note that since decay rates of size 1 jobs are infinite, they are omitted.

the *derived asymptotic delay decay rate of FB is accurate for realistic settings* where web-servers and routers are typically accessed by much more than 20 simultaneous flows.

In Figures 10 and 11, we investigate the behavior of the decay rate of FB as m and M are varied, respectively, for high load ($\rho = 0.8$). Note that we have also investigated other loads, to verify that $\rho = 0.8$ is a representative choice.

These plots illustrate the effect of increasing m and M on the decay rate of delay. Figure 10 shows that as the threshold value m increases, the decay rate increases, and thus $Pr(W(k) > m)$ decreases. As the maximum job size M increases, Figure 11 shows that the decay rate of all job sizes decreases. Both figures show that FB behaves similarly as SMART-LD for varying m and M.

Finally, observe that (as in the case of SMART-LD) small job sizes have much better decay rates under FB policies than under FCFS; whereas large job sizes have better decay rates under FCFS than under FB. Again, there is always crossover point for FB where the decay rate "crosses over" that of FCFS, and this crossover point is highly dependent on the service distribution and load.

6 A comparison of SMART-LD, FB, and PS

Now that we have derived the decay rates of SMART-LD policies and FB, it is interesting to look more closely at the behavior of these decay rates. In Sections 4 and 5, we began to explore the behavior of these decay rates, but we will now contrast their behavior. In particular, our goal will be to answer the following two practical questions.

First, *we would like to understand how much penalty FB pays for not using job size information to prioritize.* That is, by how much do SMART-LD policies outperform FB? To answer this question, we prove that the decay rate of SMART-LD is uniformly better than that of FB across all job size. Further, we provide numerical results illustrating the magnitude of the differences between SMART-LD and FB.

Second, at this point we have compared SMART-LD and FB with only FCFS. However, as we discussed in the introduction, many traditional designs in computer systems use PS, e.g., in web servers and routers. Thus, *we would like to understand how the decay rates of SMART-LD and FB compare with those of PS*. Until very recently, the analysis of PS in the many sources regime was not possible. However, there is a very recent result under submission that we can use to perform the comparison [35].

The remainder of this section is organized as follows. We will first provide an analytic comparison of SMART-LD and FB. Then, we will use numerical experiments to compare SMART-LD, FB, and PS.

6.1 Analytical comparison

For ease of understanding, we first consider the simple case when jobs are one of two sizes (1 or M) and compare the delay decay rate of SMART-LD and FB. For this case, we can provide simplified expressions which relate the delay distributions of jobs to the arrival process statistics following from Theorem 1 as

$$\Pr(W^{(N)}(k) > m) \propto \max_{T \geq 0} \Pr(E_k^{(N)}(T) > C(T + m + 1)),$$

where

$$E_k^{(N)}(T) \begin{cases} A_1^N(-T, 0), & k = 1; \\ A_1^N(-T, m) + MA_M^N(-T, 0), & k = M. \end{cases}$$

Note that for size 1 jobs, above is the "best possible" delay distribution that can be achieved over the class of all work conserving policies. For FB, it follows from Theorem 2 that

$$\Pr(W^{(N)}(k) > m) \propto \max_{T \geq 0} \Pr(F_k^{(N)}(T) > C(T + m + 1)),$$

where

$$F_k^{(N)}(T) = \begin{cases} A_1^N(-T, 0), & k = 1; \\ A_1^N(-T, m) + MA_M^N(-T, 0) & \\ \quad +(M-1)A_M^N(1, m), & k = M. \end{cases}$$

Note, that for size 1 jobs, SMART-LD and FB are asymptotically identical (i.e., the decay rates are the same). On the other-hand, for size M jobs, observe that for each fixed

T, $F_M^{(N)}(T) \geq E_M^{(N)}(T)$, which immediately implies the delay of a job of original size M with FB stochastically dominates the corresponding delay with SMART-LD. Thus, the delay experienced by a size M job under FB is larger than that under SMART-LD (in distribution). Thus it follows that for the case where arriving jobs are one of two sizes (1 or M), FB is uniformly worse than any SMART-LD policy.

To extend this argument to the general case we simply note that the volume of higher priority jobs for SMART-LD is always greater than that of FB while the available capacity is the same for both. In particular, $\mathfrak{A}_{<k}(\mathbf{z}) + \mathfrak{A}_k$ for SMART-LD is always greater than $\mathfrak{A}_{<k}(\mathbf{y}) + \mathfrak{A}_k(\mathbf{y}) + \mathfrak{A}_{>k}(\mathbf{y})$ for FB (see Theorem 1 and Theorem 2). With this observation it is straightforward to attain the following corollary.

Corollary 1 *Any SMART-LD policy is uniformly better (for any job size) than the FB policy with respect to delay in the many sources large deviation regime, i.e.,*

$$I_{\overline{W}}(k,m) \geq I_{\hat{W}}(k,m),$$

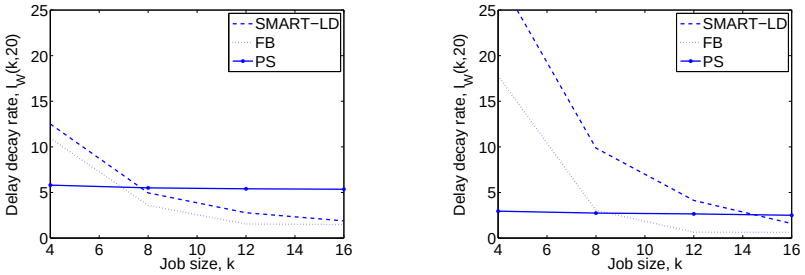
where $I_{\overline{W}}(k,m)$ and $I_{\hat{W}}(k,m)$ are the delay decay rates of all SMART-LD policies and FB respectively.

The fact that SMART-LD is better for small jobs follows from the operation of two policies: small jobs typically do not get preempted under SMART-LD, but are preempted by all arrivals under FB. However, the result that SMART-LD is better than FB even for larger jobs is less intuitive. An explanation for this fact is that under FB, though larger jobs gain higher priority at arrival compared to SMART-LD, as large jobs receive service their priority is dropping quickly under FB but may be increasing under SMART-LD.

6.2 Numerical comparison

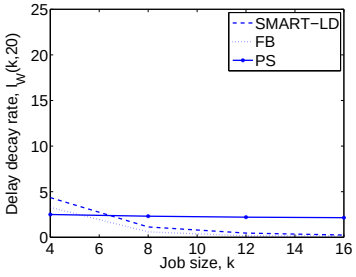
We will now move from an analytic comparison of SMART-LD and FB to a numerical comparison. This will allow us to understand the magnitude of the improvement that SMART-LD makes over FB. Further, we will take advantage of recent results characterizing the delay decay rate under PS to compare both SMART-LD and FB to PS. This comparison is important in practice because PS is often the status quo in computer systems, and thus the baseline for characterizing the improvements possible by moving to either a SMART-LD or FB based design.

Figure 12 describes how the delay decay rate for a job of size k varies across k under SMART-LD policies, FB, and PS. The results are shown for the power-law and high variability workloads under high and low loads. Note that $I_W(k,m)$ measures the decay rate of $Pr(W(k) > m)$ and that a larger $I_W(k,m)$ indicates a stochastically smaller delay. We make the following observations. First, for small job sizes, SMART-LD and FB have much better delay decay rate compared to PS. And inversely, for larger job sizes PS is better than SMART-LD and FB. Second, the figure also shows that there is always a crossover point for SMART-LD and FB where their delay decay rates "cross over" that of PS. Comparing Figure 12(a) and (c) to (b) and (d), we can see that the crossover point shifts significantly to the right for high variability distribution. In other words, SMART-LD and FB perform much better compared to PS when the variance of the distribution become higher. In addition, the crossover point does not seem to be affected much by the load. Third, the next observation we make from Figure 12 is that both SMART-LD and FB exhibit a similar trend in decay rate across k, with SMART-LD always providing stochastically smaller delays than FB for any

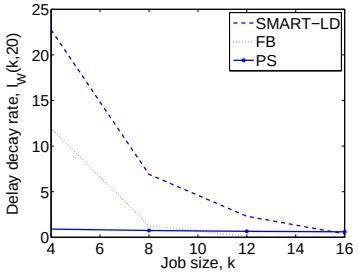


(a) Power-law (low load)

(b) High variability (low load)



(c) Power-law (high load)



(d) High variability (high load)

Fig. 12 Plot of the decay rate as a function of the job size, k, with the threshold $m = 20$ and maximum job size $M = 16$. The load is 0.2 for low load and 0.8 for high load. Note that since decay rates of size 1 jobs for both SMART-LD and FB are infinite, they are omitted.

job size. This numerical result reinforces Corollary 1 in the previous subsection. In addition, Figure 12 illustrates that the improvement of SMART-LD over FB is again highly dependent on the job size distribution: as the load of the largest jobs increases, the difference in the decay rates of SMART-LD and FB increases.

7 Conclusion

We conclude by summarizing our results. In this paper, we introduce the SMART-LD classification, which extends the SMART classification. We then prove that all SMART-LD policies are asymptotically equivalent in the many sources large deviations regime. Further, we derive expressions for the delay decay rate of the SMART-LD class and FB in the many sources regime, and we show that the delay of the SMART-LD class is always stochastically better than FB for any job size. In addition, we use numerical results to illustrate that the magnitude of this difference is highly dependent on the variability of the job size distribution – the difference increases as the variability increases. Perhaps more importantly, the decay rates we derive for the SMART-LD class and FB provide insight into how the delay occurs by identifying the "dominant" event that leads to the delay.

From a methodological point of view, the key contribution of this paper is the introduction of the 2DQ framework which enables: *(i)* the study of policies that depend on the job state (age and/or remaining size), as opposed to the queue length; and *(ii)* the study of a *class of policies*, as opposed to the analysis of individual policies. The first point allows the analysis of a large number of scheduling policies that were not feasible due to their complex behavior when a large number of flows access the system. The second point is also

important since practitioners can not implement the idealized policies in most cases, such as SRPT, that are typically the focus of theoretical research. However, by analyzing classes of policies defined by scheduling heuristics, theoretical results can hopefully be applied to the practical variations that are actually implemented. We believe that the proposed 2DQ framework that we introduce provides a promising approach for the analysis of other priority based scheduling policies and classes of policies in the many sources large deviations regime – a regime of increasing relevance to high traffic modern computer systems such as web servers and routers.

References

1. M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.
2. D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis. Asymptotic buffer overflow probabilities in multiclass multiplexers: An optimal control approach. *IEEE Trans. on Auto. Control*, 43:315–335, 1998.
3. S. C. Borst, O. J. Boxma, R. Nunez-Queija, and B. Zwart. The impact of the service discipline on delay asymptotics. *Performance Evaluation*, 54(2):175–206, October 2003.
4. D. Botvich and N. Duffield. Large deviations, economies of scale, and the shape of the loss curve in large multiplexers. *Queueing Systems*, 20:293–320, 1995.
5. M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.
6. S. Delas, R. Mazumdar, and C. Rosenberg. Cell loss asymptotics for buffers handling a large number of independent stationary sources. In *Proc of IEEE Infocom*, volume 2, pages 551–558, 1999.
7. S. Delas, R. Mazumdar, and C. Rosenberg. Tail asymptotics for HOL priority queues handling a large number of independent stationary sources. *Queue. Sys. Thry. and App.*, 40(2):183–204, 2002.
8. H. Feng, V. Misra, and D. Rubenstein. PBS: a unified priority-based cpu scheduler. In *Proc. of ACM Sigmetrics*, 2007.
9. M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Implementation of SRPT scheduling in web servers. *ACM Trans. on Comp. Sys.*, 21(2), May 2003.
10. M. Hu, J. Zhang, and J. Sadowsky. A size-aided opportunistic scheduling scheme in wireless networks. In *Globecom*, 2003.
11. C. Kotopoulos, N. Likhanov, and R. Mazumdar. Overflow asymptotics in GPS systems with heterogeneous longtailed inputs. In *Proc. of IEEE Infocom*, 2001.
12. N. Likhanov and R. Mazumdar. Cell loss asymptotics for buffers fed with a large number of independent stationary sources. *Journal of Applied Probability*, 36:86–96, 1999.
13. D. Lu, P. Dinda, Y. Qiao, and H. Sheng. Effects and implications of file size/service time correlation on web server scheduling policies. In *Proc. of IEEE Mascots*, 2005.
14. M. Mandjes and M. Nuyens. Sojourn time in the M/G/1 FB queue with light-tailed service times. *Prob. in the Eng. and Info. Sci.*, 19:351–361, 2005.
15. L. Massoulie. Large deviations estimates for polling and weighted fair queueing service systems. *Adv. Perf. Anal.*, 2(2):103–128, 1999.
16. R. Nunez-Queija. Queues with equally heavy sojourn time and service requirement distributions. *Ann. Oper. Res.*, 113:101–117, 2002.
17. M. Nuyens. The foreground-background queue. *PhD thesis, University of Amsterdam*, 2004.
18. M. Nuyens and A. Wierman. The foreground-background queue: A survey. *Performance Evaluation*, 65(3-4):286–307, 2008.
19. M. Nuyens, A. Wierman, and B. Zwart. Preventing large sojourn times using SMART scheduling. *Operations Research*, 56(1):88–101, 2008.
20. M. Nuyens and B. Zwart. A large-deviation analysis of the GI/GI/1 SRPT queue. *Queueing Systems*, 54(2):85–97, 2006.
21. Y. Qiao, D. Lu, R. Bustamante, and P. Dinda. Looking at the server side of peer-to-peer systems. Technical Report NWU-CS-04-37, Northwestern University, 2004.
22. I. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of LAS scheduling for job size distributions with high variance. In *Proc. of ACM Sigmetrics*, 2003.
23. I. Rai, G. Urvoy-Keller, M. Vernon, and E. Biersack. Performance modeling of LAS based scheduling in packet switched networks. In *Proc. of ACM Sigmetrics/Performance*, 2004.

24. M. Rawat and A. Kshemkalyani. SWIFT: Scheduling in web servers for fast response time. In *Symp. on Net. Comp. and App.*, 2003.
25. R. Righter and J. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Prob. in the Eng. and Info. Sci.*, 3:967–978, 1989.
26. R. Righter, J. Shanthikumar, and G. Yamazaki. On external service disciplines in single stage queueing systems. *Journal of Applied Probability*, 27:409–416, 1990.
27. L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.
28. B. Schroeder and M. Harchol-Balter. Web servers under overload: How scheduling can help. *ACM Transactions on Internet Technology*, 6(1), 2006.
29. A. Schwartz and A. Weiss. *Large deviations for performance analysis*. Chapman and Hall, London, UK, 1995.
30. S. Shakkottai and R. Srikant. Many-sources delay asymptotics with applications to priority queues. *Queueing Systems: Theory and Applications*, 39:183–200, October 2001.
31. A. Wierman. Fairness and classifications. *Performance Evaluation Review*, 34(4):4–12, 2007.
32. A. Wierman. *Scheduling for today's computer systems: Bridging theory and practice*. PhD thesis, Carnegie Mellon University, 2007.
33. A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proc. of ACM Sigmetrics*, 2003.
34. A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *Proc. of ACM Sigmetrics*, 2005.
35. C. W. Yang and S. Shakkottai. A discrete version of processor sharing accessed by a large number of flows. *Under Submission*.
36. C. W. Yang and S. Shakkottai. Delay asymptote of the SRPT scheduler. In *Proceedings of the IEEE Conference on Decision and Control*, December 2004.