Fairness and Classifications

Adam Wierman
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15217
acw@cs.cmu.edu

ABSTRACT

The growing trend in computer systems towards using scheduling policies that prioritize jobs with small service requirements has resulted in a new focus on the fairness of such policies. In particular, researchers have been interested in whether prioritizing small job sizes results in large jobs being treated "unfairly." However, fairness is an amorphous concept and thus difficult to define and study. This article provides a short survey of recent work in this area.

1. INTRODUCTION

Traditionally, the performance of scheduling policies has been measured using mean response time (a.k.a. sojourn time, flow time), and more recently mean slowdown [3, 16, 24] and the tail of response time [6, 7, 26]. Under these measures, policies that give priority to small job sizes (a.k.a. service requirements) at the expense of larger job sizes perform quite well. For example, Shortest-Remaining-Processing-Time (SRPT) is known to optimize mean response time [36, 37]. As a result, designs based on these policies have been suggested for a variety of computer systems in recent years. However the adoption of these new designs has been slow due to fears about the fairness of these policies. Specifically, there are worries that large job sizes may be "starved" of service under a policy that gives priority to small job sizes, which would result in large job sizes having response times that are unfairly long and variable [5, 38, 39, 40].

These worries have recurred nearly everywhere size based policies have been suggested. A first example is the case of web servers, where recent designs have illustrated that giving priority to requests for small files can significantly reduce response times [17, 29]. However, it is important that this improvement does not come at the expense of providing large job sizes unfairly large response times, which are typically associated with the important requests. It is in this setting that Bansal & Harchol-Balter provided the first study of the fairness of SRPT [4]. The same tradeoff has appeared across diverse application areas. For example, UNIX processes are assigned decreasing priority based on their current age – CPU usage so far. The worry is that this may create unfairness for old processes [10]. Similar tradeoffs can be found in recent designs for routers [27, 28], wireless networks [19], transport protocols [45], and beyond.

To address these worries, there has been an explosion of theoretical work studying the "fairness" of priority-based policies, and it is the purpose of this paper to present a brief survey of this work.

Before moving forward however, let us take a moment to define the model we are considering and the notation we will use. Our focus will be on work conserving, preempt-resume single server queues, *the M/GI/1 model unless otherwise stated.* The policies that we will consider are summarized in Table 1. We define T and $T(x)$ to be the steady-state response time overall and for a job of size x respectively, and we let $\rho < 1$ be the system load. That is $\rho = \lambda E[X]$, where λ is the arrival rate of the system and X is a random variable distributed according to a continuous service (job size) distribution $F(x)$ having density function $f(x)$ defined for all $x \geq 0$. Let $\overline{F}(x) = 1 - F(x)$. Further, define $\rho(x) = \lambda F(x) E[X | X < x]$ and $\widetilde{\rho}(x) = \lambda E[\min(X, x)]$. Finally, let B denote the length of a busy period.

2. DEFINING FAIRNESS

Fairness is an amorphous concept, and nearly impossible to define in a universal way. The difficulty in defining the fairness of scheduling policies is best illustrated using a few simple examples:

(i) Suppose jobs a and b are the same size, and a enters the queue slightly before b.

(ii) Suppose jobs c and d are very large and very small respectively, and job c enters the queue slightly before job d.

Most people agree that it is fair to serve job a before job b and to serve job d before job c. However, notice how the fair service order changes depending on the setting being considered. If the queue in question is a ticket box office, then it is more fair to serve job c before job d and if the setting is a hospital it is more fair to serve the more urgent job, regardless of the sizes or arrival order.

This simple example illustrates that we must have a clear idea of what is meant by the term "fair" in the application context before defining an appropriate notion of fairness. In this survey, our focus is on evaluating the fairness of system designs that propose to give priority to small job sizes. *Thus, in our context "fairness" refers to the idea that all job sizes should receive equitable service, i.e. no job size receives unduely large response times.* It is important to realize that this is only one possible meaning of the term "fairness," and in other applications fairness could mean something completely different. In Section 7, we will briefly discuss this topic further.

Moving forward, let us reiterate that our goal is to evaluate the fairness of priority-based scheduling (particularly the heuristic of giving priority to small jobs) with respect to the performance of different job sizes (particularly large jobs). Thus, we are interested in understanding the behavior of $T(x)$ across x.

In order to compare the response times of different job sizes, we need to first understand how to normalize the response times in order to make the comparison, i.e. how to normalize $T(x)$ in order to compare it across x. In addition, it is important to define a criterion that indicates when the normalized response time is considered "fair." The following metric and criterion were proposed by Wierman & Harchol-Balter in [41]:

`FB`	Foreground-Background preemptively serves those jobs that have received the least amount of service so far.
`FCFS`	First Come First Served serves jobs in the order they arrive.
`LCFS`	Last Come First Served non-preemptively serves the job that arrived the most recently.
`LRPT`	Longest Remaining Processing Time preemptively serves the job in the system with the largest remaining processing time.
`LJF`	Longest Job First non-preemptively serves the job in the system with the largest original size.
`PLCFS`	Preemptive Last Come First Served preemptively serves the most recent arrival.
`PLJF`	Preemptive Longest Job First preemptively serves the job in the system with the largest original size.
`PS`	Processor Sharing serves all customers simultaneously, at the same rate.
`PSJF`	Preemptive Shortest Job First preemptively serves the job in the system with the smallest original size.
`SJF`	Shortest Job First non-preemptively serves the job in the system with the smallest original size.
`SRPT`	Shortest Remaining Processing Time preemptively serves the job with the shortest remaining size.

Table 1: ***A brief description of the scheduling policies discussed in this paper.***

DEFINITION 1. *Let $0 < \rho < 1$ in an M/GI/1. A job size x is treated* ***fairly*** *under policy P, service distribution X, and load ρ if*

$$\frac{E[T(x)]^P}{x} \leq \frac{1}{1-\rho}$$

Otherwise a job size x is treated ***unfairly****. A scheduling policy P is* ***fair*** *if every job size is treated fairly. Otherwise P is* ***unfair****.*

Definition 1 consists of two pieces: a metric, $E[T(x)]/x$, and a criterion, $1/(1-\rho)$. Each of these pieces can be motivated in a variety of ways.

Let us first consider some justifications for the metric $E[T(x)]/x$. The metric $E[T(x)]/x$, which is referred to as the *mean conditional slowdown*, derives intuitively from Aristotle's notion of fairness: like cases should be treated alike; different cases should be treated differently; and different cases should be treated differently in proportion to the difference at stake [33]. In the context of scheduling queues, this matches the common intuition that: small jobs should have small response times, large jobs should have large response times, and the differences in response times of small and large jobs should be proportional to the differences between the job sizes. More formally, when comparing $E[T(x)]^P$ across x, we want a *metric* that scales $E[T(x)]^P$ appropriately to allow for comparison of $E[T(x)]^P$ between small and large x. For $E[T(x)]^P$, it is clear that $1/x$ is an appropriate scaling factor because $E[T(x)]^P = \Theta(x)$ under all work conserving scheduling policies (see Theorem 10), and thus we need to normalize by $1/x$.

To justify the criterion $1/(1-\rho)$, we will again start with an intuitive justification. `PS` is typically thought of as a fair policy because at every instant every job in the system receives an equal share of the server. This also corresponds to Rawls' theory of social justice [30]. Further, `PS` satisfies the idea that $E[T(x)]$ should be proportional to x, since $E[T(x)]^{PS}/x = 1/(1-\rho)$. Thus, $1/(1-\rho)$ is a natural choice for a fairness criterion. In addition, it is a practical choice because in many computer applications `PS` is the status quo, and by comparing with the status quo we develop an understanding of how new designs will affect response times. In addition to these justifications, we can also provide a more formal justification of the criterion $1/(1-\rho)$. We can view it as a form of min-max fairness (a.k.a. Pareto efficiency). In our context, Wierman & Harchol-Balter [41] prove that $\min_P \max_x E[T(x)]^P/x = 1/(1-\rho)$, and thus $1/(1-\rho)$ is the appropriate bar for comparison. Finally, it is important to point out that $1/(1-\rho)$ will turn out to be a useful criterion because it distinguishes between distinct patterns of behavior of policies with respect to $E[T(x)]^P/x$ (see Figure 1).

With Definition 1 in hand, it is possible to group scheduling policies based on whether they (i) treat all job sizes fairly or (ii) treat some job sizes unfairly. Curiously, some policies may fall into either type (i) or type (ii) depending on the load and service distribution. We therefore define *three types of unfairness*.

DEFINITION 2. *Let $0 < \rho < 1$ in an M/GI/1 queue. A scheduling policy P is: (i)* ***Always Fair*** *if P is fair for all such ρ and X; (ii)* ***Sometimes Fair*** *if P is fair under some ρ and X and unfair under other ρ and X; or (iii)* ***Always Unfair*** *if P is unfair under all loads and service distributions.*

Note that Definition 2 is a generalization of the definition in [41]. Initially, in [41], only distributions with finite variance were considered. However, more recently Brown [8] was able to extend many of the results to allow the consideration of distributions with *infinite* variance. The extension led to some counter-intuitive results that we will discuss later in the paper.

3. STUDYING COMMON POLICIES

The purpose of Definition 1 is to be able to understand the fairness of recent system designs that propose policies that prioritize small jobs. To that end, in this section we will survey recent results characterizing the fairness of two important policies: `SRPT` and `FB`. Designs based on `SRPT` have been proposed for use in a number of application areas, including web servers [17, 29] and wireless networks [19], and designs based on `FB` have been proposed for use in operating systems [10] and routers [27, 28].

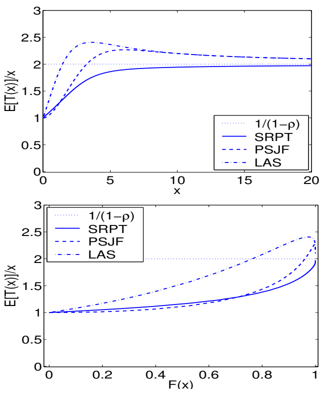
However, before we can study `SRPT` and `FB`, it is useful to develop an understanding of Definition 1 by studying the fairness of a few simpler policies. As a first step, it is easy to see that `PS` and `PLCFS` are Always Fair because

$$E[T(x)]^{PLCFS} = E[T(x)]^{PS} = \frac{x}{1-\rho}$$

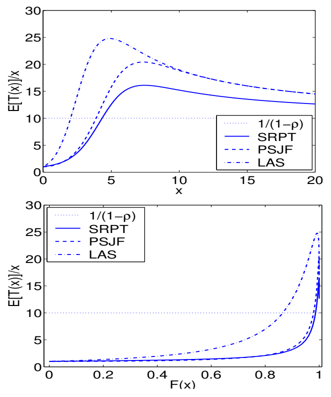
Thus, there are a number of policies that are fair under all service distributions and all loads.

However, most common policies are not Always Fair. In fact, most are Always Unfair. For example, it is straightforward to see that `FCFS` is Always Unfair, since the smallest job size in the service distribution will be treated unfairly [41]. But, it turns out that by giving priority to small jobs, it is possible to be Sometimes Fair. In particular, we will see that `SRPT` and `FB` are Sometimes Fair. In fact, we will see that even in the settings where these policies are unfair, the degree of unfairness to large job sizes is not as bad as one might expect.

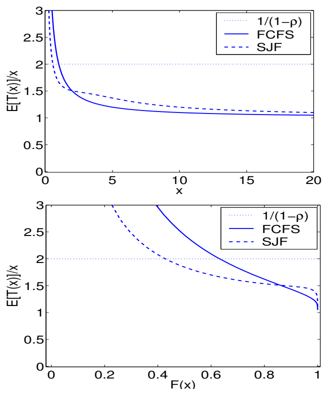
We will now delve into detail about the fairness of `SRPT` and `FB`. Note though that although we only consider `SRPT` and `FB` in detail, the fairness of a wide variety of other common policies have been analyzed in [4, 8, 13, 18, 27, 28, 41]. The reader may find illustrations of the fairness behavior of `FB`, `SRPT`, and many other common policies in Figures 1 and 4.



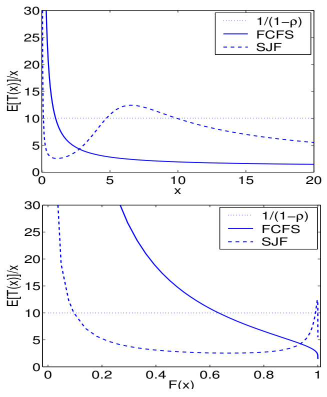
(a) Low load ($\rho = 0.5$)



(b) High load ($\rho = 0.9$)



(c) Low load ($\rho = 0.5$)



(d) High load ($\rho = 0.9$)

Figure 1: The conditional mean slowdown is illustrated under a variety of both preemptive and non-preemptive policies. The service distribution is exponential with mean 1. The dotted line shows the criterion for fairness.

3.1 SRPT

SRPT has long been known to optimize $E[T]$. However, its use in practice has been hindered by the fear that large job sizes experience unfairly long response times. The fairness of SRPT was first studied by Bansal & Harchol-Balter [4] under the assumption that $E[X^2] < \infty$. These initial results were later extended by Wierman & Harchol-Balter [41], and then by Brown [8], who was the first to consider fairness when $E[X^2] = \infty$. We summarize the major results in the following theorem:

THEOREM 1. SRPT *is Sometimes Fair.* SRPT *is fair when* $\rho \leq 0.5$ *or when the service distribution is regularly varying*[1] *with rate* $\alpha \in (1, 1.5)$. *However, when* $E[X^2] < \infty$, *under all service distributions there exists a* $\rho_c < 1$ *such that for all* $\rho > \rho_c$ SRPT *is unfair.*

Theorem 1 illustrates that, surprisingly, in many cases SRPT is fair to all job sizes. Thus, in many cases it is possible to optimize $E[T]$ while still providing fair response times to all job sizes. In particular, when the system load is small enough or the tail of the service distribution is heavy enough, SRPT is fair. In fact, SRPT is often fair in practical settings since workloads in many computer applications are thought to be regularly varying with $\alpha \in (1, 1.5)$.

However, SRPT is not Always Fair, because when $E[X^2] < \infty$, under high enough load SRPT becomes unfair. This behavior is illustrated in Figure 1. Note that an important open question is determining the ρ_c below which SRPT is fair and above which SRPT is unfair. This ρ_c is highly service distribution dependent, and currently all that is known is that $\rho_c \in (0.5, 1)$.

Though SRPT is fair in many cases, there are also many situations when SRPT is unfair and, in these cases, it is important to understand which job sizes are treated unfairly and how unfairly these job sizes are treated. The following theorem summarizes the results from [4, 41]:

THEOREM 2. *All* x *such that* $\rho(x) \leq \max(1/2, 2\rho/3, 1-\sqrt{1-\rho})$ *are treated fairly under* SRPT. *Further, for any* x *that is treated unfairly,*

$$E[T(x)]^{SRPT} \leq \left(\frac{2-\rho}{2(1-\rho)}\right)\frac{1}{1-\rho}.$$

[1]Regularly varying distributions are a class of distributions that have the same tail behavior as the Pareto. Formally, a service distribution is regularly varying if $\overline{F}(x) = L(x)x^{-\alpha}$ for some $L(x)$ such that for all $y > 0$, $L(yx)/L(x) \to 1$ as $x \to \infty$.

Notice that only a small fraction of jobs are treated unfairly and that these jobs experience a quite small degree of unfairness. Further, it is important to note that as ρ increases, so does the lower bound $1 - \sqrt{1-\rho}$ on $\rho(x)$. In fact, this bound converges to 1 as $\rho \to 1$, which signifies that the size of the smallest job that might be treated unfairly is increasing unboundedly as ρ increases.

One expects that when SRPT is unfair, it is unfair to large jobs and most unfair to the largest ones. It turns out that SRPT is unfair to (a small fraction of) large jobs, but that it is most unfair to some range medium-large jobs. Specifically, as $x \to \infty$, $E[T(x)]/x \to 1/(1-\rho)$ and the limit is approached from above [41]. This behavior is illustrated in Figure 1.

In summary, it seems that worries about the starvation of large job sizes under SRPT can safely be discarded since in many practical settings SRPT is fair. Further, when SRPT is unfair, only a small fraction of jobs are treated unfairly and the degree of unfairness these jobs experience is small.

3.2 FB

FB[2] has long been known to optimize $E[T]$ among policies blind to job size information when job sizes have decreasing failure rate [34]. However, like SRPT, its use in practice has long been hindered by the fear that large job sizes experience unfairly large response times. The first results on the fairness of FB were published simultaneously by Wierman & Harchol-Balter [41] and Rai, Urvoy-Keller, & Biersack [27] under the assumption that $E[X^2] < \infty$. However, recently Brown found, surprisingly, that FB can be fair when the service distribution has infinite variance [8]. We summarize the major results in the following theorem:

THEOREM 3. FB *is Sometimes Fair.* FB *is unfair when* $E[X^2] < \infty$. *However,* FB *is fair when the service distribution is regularly varying with rate* $\alpha \in (1, 1.5)$.

Comparing Theorem 3 with Theorem 1 indicates that FB is more unfair than SRPT when $E[X^2] < \infty$ but that FB parallels the fairness of SRPT under regularly varying distributions. Intuitively, this is a result of the fact that, the heavier the tail of the service distribution, the closer to SRPT FB becomes. In particular, it is well known that under regularly varying service distributions FB behaves very

[2]FB is known by a variety of other names including Least Attained Service (LAS) and Shortest Elapsed Time first (SET). See [25] for a survey.

similarly to SRPT with respect to $E[T]$ and $P(T > x)$ [26, 25]. Theorem 3 indicates that this similarity also extends to fairness.

Though Theorem 3 says that FB can be fair in some situations, more often than not, FB is unfair. Thus, it is important to understand which job sizes FB is unfair to and how unfairly these job sizes are treated. A number of researchers have looked into this question. Summarizing the major results from [8, 27, 41], we have the following answers to these questions:

THEOREM 4. *All x such that $\widetilde{\rho}(x) \leq \max(2\rho/3, 1-\sqrt{1-\rho})$ are treated fairly under* FB. *Further, for any x that is treated unfairly,*

$$E[T(x)]^{FB} \leq \left(\frac{2-\rho}{2(1-\rho)}\right)\frac{1}{1-\rho}.$$

Notice that the percentage of job sizes that are treated unfairly is quite small in most settings, and especially in the case when the service distribution is highly variable since under such distributions a small percentage of the largest jobs make up a large fraction of the load. Further, notice that the degree of unfairness experienced by this small fraction of jobs is bounded independently of the service distribution, and is small when the system is not heavily loaded.

Like with SRPT, one expects that when FB is unfair, it is unfair to large jobs and most unfair to the largest ones. It turns out that FB is indeed unfair to (a small fraction) of large jobs, but that it is most unfair to some range medium-large jobs. Specifically, as $x \to \infty$, $E[T(x)]^{FB}/x \to 1/(1-\rho)$ and the limit is approached from above [41]. This behavior is illustrated in Figure 1.

In summary, the results in this section should serve to ease concerns about the unfairness experienced by large job sizes under FB. When FB is unfair, it is to a very small fraction of jobs and the degree of unfairness experienced by these jobs is small. Further, in many practical settings FB is fair to all job sizes.

4. STUDYING SCHEDULING CLASSIFICATIONS

Traditionally, research studying the scheduling of queues has focused on a limited range of idealized scheduling policies, as we have in this paper to this point. However these policies are hardly ever implemented in their pure form in computer systems. For example, though many recent systems have been designed using the heuristic of "prioritizing small jobs" none have implemented pure SRPT or FB as a result of limitations on the scheduling techniques that are possible and the desire to optimize a range of secondary metrics. Instead hybrid policies have been proposed, e.g. [14, 15, 28, 10]. Thus, there is a gap between the results provided by traditional research and the needs of practitioners.

An emerging style of research attempts to bridge this gap by formalizing general *scheduling heuristics* (such as giving priority to small/large jobs) and *scheduling techniques* (such as prioritizing using remaining sizes), and then analyzing these heuristics and techniques instead of analyzing the behavior of idealized individual policies [22, 10, 41, 42, 43]. The analysis of these heuristic classifications provides both practical and theoretical benefits. Theoretically, such results add structure to the space of scheduling policies that cannot be attained by studying individual policies alone. Practically, such results provide analyses of the hybrid policies that are implemented in real systems.

4.1 Scheduling heuristics

The goal of this section is to characterize the fairness of three common scheduling heuristics: *prioritizing small jobs, prioritizing large jobs, and symmetric scheduling*. The scheduling heuristic employed by a policy is a key factor in determining the performance of the policy. For example, though recent web server designs have been motivated by the optimality of SRPT, pure SRPT is not implemented. However, the policies that are implemented do obey the heuristic of prioritizing small jobs, and therefore still provide near optimal mean response time.

4.1.1 Prioritizing small jobs

It is well known that policies that "prioritize small jobs" perform well with respect to mean response time. 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. The SMART class, recently introduced by Wierman, Harchol-Balter & Osogami [43] formalizes the heuristic of "prioritizing small jobs" in order to provide "SMAll Response Times[3]" using three simple properties described below.

We denote the remaining size, original size, and arrival time of job a as r_a, s_a, and t_a respectively. The original sizes, remaining sizes, and arrival times of b and c are defined similarly. We define *job a to have priority over job b* if job b can never run while job a is in the system.

DEFINITION 3. *A work conserving policy P belongs to the* SMART *class, denoted $P \in$* SMART*, if it obeys the following properties.*

Bias Property: *If $r_b > s_a$, then job a has priority over job b.*

Consistency Property: *If job a ever receives service while job b is in the system, thereafter job a has priority over job b.*

Transitivity Property: *If an arriving job b preempts job c at time t; thereafter, until job c receives service, every arrival, a, with size $s_a < s_b$ is given priority over job c.*

This definition has been crafted to mimic the heuristic of biasing towards jobs that are (originally) short or have small remaining service requirements. The Bias Property guarantees that the job being run at the server has remaining size smaller than the original size of all jobs in the system. The Consistency and Transitivity Properties ensure *coherency* in the priority structure enforced by the Bias Property. In particular, the Consistency Property prevents time-sharing by guaranteeing that after job a is chosen to run ahead of b, job b will never run ahead of job a. This makes intuitive sense because our priority function is based on the heuristic of giving priority to small jobs, and as job a receives service, it can only get smaller. Finally, the Transitivity Property guarantees that SMART policies do not second guess themselves: if an arrival a is estimated to be "smaller" than job b (and hence is given priority over job b), future arrivals smaller than a are also considered "smaller" than b.

Many common policies are part of the SMART class, e.g. SRPT and PSJF. Further, the SMART class includes many generalizations of these policies, including policies that use time-varying priority functions (see [26] for details).

Over the past few years, a number of results have been proven about SMART policies [26, 43, 44]. Of interest to us here are the bounds that have been proven by Wierman, Harchol-Balter, & Osogami on $E[T]$ and $T(x)$ [43].

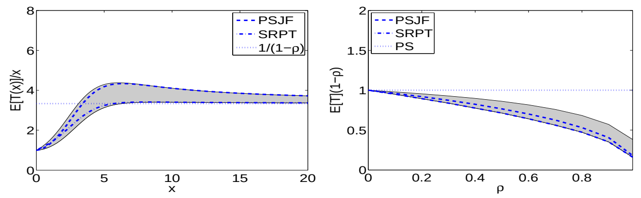
THEOREM 5. *For $P \in$* SMART*:*

$$R(x)^{SRPT}+W(x)^{PSJF} \leq_{st} T(x)^P \leq_{st} R(x)^{PSJF}+W(x)^{SRPT}.$$

It follows that $T(x)^P \leq_{st} T(x)^{FB}$. Further,

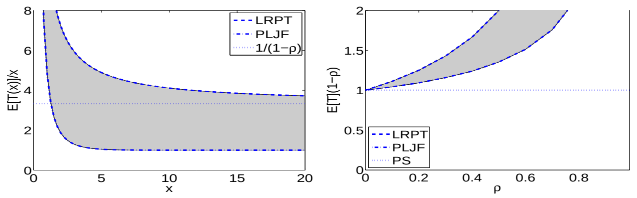
$$E[T]^{SRPT} \leq E[T]^P \leq 2E[T]^{SRPT}$$

[3]We thank Hanoch Levy for his suggestion of this acronym.



(a) $E[T(x)]/x$ (b) $E[T](1-\rho)$

Figure 2: This figure illustrates the bounds on $E[T(x)]/x$ and $E[T]$ under the class of `SMART` policies. The shaded area indicates the response times attainable using `SMART` policies. In addition, the behaviors of the two most common `SMART` policies, `SRPT` and `PSJF`, are illustrated and compared with `PS`. In both plots the service distribution is taken to be Exponential with mean 1, and the load in (a) is 0.7.



(a) $E[T(x)]$ (b) $E[T](1-\rho)$

Figure 3: This figure illustrates the bounds on $E[T(x)]/x$ and $E[T]$ under the class of `FOOLISH` policies. The shaded area indicates the response times attainable using `FOOLISH` policies. In addition, the behaviors of the two most common `FOOLISH` policies, `PLJF` and `LRPT`, are illustrated and compared with `PS`. The service distribution is taken to be Exponential with mean 1, and the load in (a) is 0.7.

This result (illustrated in Figure 2) serves as a validation of the heuristic of prioritizing small jobs, since it indicates that all policies that obey this heuristic have near optimal response times. Further, the fairness properties of the class follow from Theorem 5 using the same techniques as used in [41] to analyze `SRPT`.

THEOREM 6. *All `SMART` policies are Sometimes Fair.*

This theorem is in many ways a negative result, since it indicates that no policy that prioritizes small jobs (and thus has near optimal $E[T]$) can be Always Fair. However, it is also a positive result in the sense that the fairness of these policies is not as bad as expected, and there are practical settings where they are fair.

4.1.2 Prioritizing large jobs

We now move from policies that prioritize small jobs to policies that prioritize large jobs, which we term `FOOLISH` policies. Though `FOOLISH` policies may not be practical in many settings, we introduce them here in order to contrast the behavior of policies that bias towards large jobs with the behavior of `SMART` policies.

DEFINITION 4. *A work conserving policy $P \in$ `FOOLISH` if it obeys the following property:*

Bias Property: *If $r_b > r_a$ and $s_b > s_a$, then job b has priority over job a.*

Notice that Definition 4 parallels Definition 3 of `SMART` policies. However, an important difference between the two definitions is that the definition of `FOOLISH` policies does not include a Consistency or Transitivity Property. This is because these properties are used by the `SMART` definition to avoid time-sharing and other such behaviors that increase $E[T]$. However, all `FOOLISH` policies will have large $E[T]$, so there is no need to exclude these behaviors from the `FOOLISH` class.

Many common policies are part of the `FOOLISH` class. Of course, the `FOOLISH` class includes `LRPT` and `PLJF`. Further, it is easy to prove that the `FOOLISH` class includes a range of policies having more complicated (maybe time-varying) priority schemes, similarly to the `SMART` class.

Let us now discuss $E[T]$ and $T(x)$ under `FOOLISH` policies.

THEOREM 7. *In a GI/GI/1 queue, for $P \in$ `FOOLISH`,*

$$T(x)^{PLJF} \le_{st} T(x)^{P} \le_{st} T(x)^{LRPT}.$$

Thus, $E[T]^{PLJF} \le E[T]^{P} \le E[T]^{LRPT}$.

The bounds on $E[T]$ under `FOOLISH` policies are illustrated in Figure 3. It is interesting to compare the bounds on $E[T]$ under `FOOLISH` and `SMART` policies. The mean response time under `SMART` policies is significantly smaller than the mean response time under `FOOLISH` policies, and in fact, under many service distributions (for instance the Exponential used in Figure 3) all `SMART` policies have smaller mean response time than all `FOOLISH` policies under all loads. Further, from Figure 3 it is immediately evident that `FOOLISH` policies are unfair to small job sizes. The proof of this fact follows easily from Theorem 7.

THEOREM 8. *All `FOOLISH` policies are Always Unfair.*

4.1.3 Symmetric scheduling

Neither of the scheduling heuristics we have considered so far are Always Fair. However, there is one common scheduling heuristic that is Always Fair: the class of `SYMMETRIC` scheduling policies, which was introduced by Kelly [20]. `SYMMETRIC` disciplines have proved fundamental to the study of queueing networks due to the important property that the departure process is stochastically identical to the arrival process when time is reversed under these policies. In this paper, we consider the class of `SYMMETRIC` disciplines not because of their behavior in queueing networks, but rather due to the sense of "fairness" they provide. Unlike `SMART` and `FOOLISH` policies, `SYMMETRIC` policies do not prioritize based on job size information. Instead, `SYMMETRIC` policies provide "fairness" in the sense that they do not schedule based on any job traits – all arrivals are treated equivalently.

Let us now define the `SYMMETRIC` class. Consider a queue containing customers in positions $1, 2, \ldots, n$ where upon the completion of the ith job the jobs in positions $i+1, \ldots, n$ move to positions $i, \ldots, n-1$ and upon an arrival to the ith position the jobs in position $i, \ldots, n$ move to positions $i+1, \ldots, n+1$.

DEFINITION 5. *A scheduling policy is an **M/GI/1** `SYMMETRIC` **discipline** if when n jobs are in the queue, a proportion $\delta(i,n)$ of the server is directed to the ith customer in the queue, and an arrival enters position i with probability $\delta(i,n)$. Of course, for all n, $\sum_{i=1}^{n} \delta(i,n) = 1$.*

Intuitively, `SYMMETRIC` policies are policies where the arrival and service rates for each position in the queue are "symmetric." It may not be immediately apparent, but the class of `SYMMETRIC` policies is quite broad, e.g. by taking $\delta(i,n) = 1/n$ we obtain `PS` and by taking $\delta(i,n) = 1_{[i=n]}$ we obtain `PLCFS`.

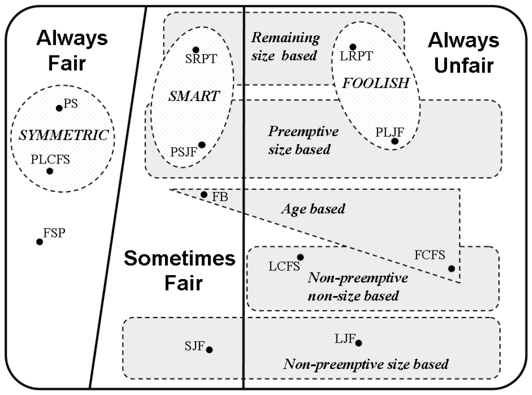


Figure 4: An illustration of the classification of common prioritization techniques and heuristics with respect to fairness.

Amazingly, all SYMMETRIC policies have equivalent mean response times, and thus all SYMMETRIC policies are Always Fair, which is a stark contrast to SMART and FOOLISH policies. However, though SYMMETRIC policies are Always Fair, they are inefficient in the sense that they have $E[T]$ far from optimal. Thus, in order to be fair they sacrifice efficiency.

THEOREM 9. *All* SYMMETRIC *policies are Always Fair. In particular, for $P \in$ SYMMETRIC,*

$$E[T(x)]^P = \frac{x}{1-\rho} \quad \textit{and thus} \quad E[T]^P = \frac{E[X]}{1-\rho}$$

4.2 Scheduling techniques

Figure 4 illustrates our results thus far with respect to both individual scheduling policies and *heuristic-based* classifications of policies. However the scheduling heuristic applied in a policy is only one defining aspect of the discipline. For instance, SRPT is defined both by the fact that it prioritizes small jobs and the fact that it uses remaining sizes to prioritize. Therefore, in addition to heuristic-based classifications, we also consider *technique-based* classifications For example, consider all scheduling policies that prioritize using purely the remaining size of a job. This includes much more than SRPT, for example it includes LRPT as well as hybrid policies that prioritize both large and small jobs in an attempt to curb unfairness. We call this the class of remaining size based policies. Similarly, we can define the class of preemptive size based policies to be all preemptive policies that make use only of the original size of a job. Figure 4 provides an overview of the fairness of these and other technique-based classifications. The reader can find more information about these technique-based classifications in [41, 42].

5. A FAIR AND EFFICIENT POLICY

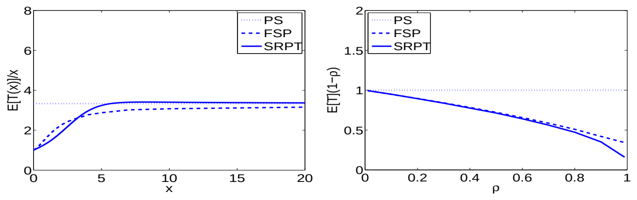
Over the last few sections, we have seen that there are very few common policies that are Always Fair, and further that those policies that are Always Fair have mean response times that can be far from optimal. In fact, the results we have presented so far paint a discouraging picture: no common scheduling heuristics and techniques can result in policies that are Always Fair and no policies can be both fair and efficient.

The search for a fair policy with near optimal performance proved elusive until the 2003 ACM Sigmetrics conference when Friedman and Henderson presented a new policy called Fair-Sojourn-Protocol (FSP), which provides the first example of a fair policy that significantly improves upon the performance of PS. [11].

FSP is motivated by one simple idea: at any given point, it is easy to tell which job will finish next under PS. Given this information, it is possible to avoid wasting time time-sharing among jobs, and thus improve the response times of PS dramatically. The easiest way to understand FSP is to imagine that at any point in time you know the full state of a virtual PS queue, with the same arrival process. (Note though that this is not actually needed for the implementation of FSP.) Under the FSP policy, the job being run is always the job that the virtual PS queue would have completed first. Thus, FSP can be thought of as performing SRPT on the remaining times of a virtual PS system.

The power of FSP comes from the following property of the policy: FSP finishes every job at least as early PS would regardless of the arrival process and service distribution. This follows from the fact that, by its definition, FSP is only reordering the work that is being done under PS so as to be more efficient; however a formal proof can be found in [11].

Unfortunately, beyond the guarantee of better performance than PS, there exists very little work analyzing the performance of FSP. Thus, in order to understand how much FSP improves response times over PS, researchers have been limited to simulation studies such as those in [11, 14, 15]. In order to summarize the key points of these studies, we will now briefly illustrate how the mean response time of FSP compares to that of PS and further, to the optimal mean response time, $E[T]^{SRPT}$. Figure 5 illustrates a simple comparison of the response times under FSP, PS, and SRPT. In Figure 5 (a) the behavior of $E[T(x)]$ under each of these policies is shown. We can see that FSP provides improvements over PS for all job sizes and that the improvements are most dramatic for small job sizes. But, the bias towards small job sizes is not as extreme under FSP as it is under SRPT. Moving to Figure 5 (b), we can see that $E[T]^{FSP}$ and $E[T]^{SRPT}$ are similar when the load is not too high: only when $\rho \to 1$ does SRPT provide dramatic gains in $E[T]$ over FSP. These results clearly illustrate the need for future work to provide a queueing analysis of FSP.



(a) $E[T(x)]/x$ (b) $E[T](1-\rho)$

Figure 5: These plots show a comparison between PS, FSP, and SRPT with respect to both $E[T(x)]/x$ and $E[T]$. The service distribution is Exponential with mean 1. The load in (a) is 0.7.

6. BEYOND FAIRNESS IN EXPECTATION

So far, we have studied fairness in *expectation*. However, worries about the experience of large job sizes under policies that prioritize small jobs are not limited to the mean. People worry that large jobs will experience both larger and *more variable* response times [5, 38, 39, 40]. Thus, it is important to develop a framework for comparing distributional properties of $T(x)$ across x, not just $E[T(x)]$. The approach that has been followed in the literature is to compare higher moments of $T(x)$ across x, however developing a framework for comparing higher moments of $T(x)$ is not straightforward. As a result, we will start by limiting our focus to the behavior of large job sizes, and then we will exploit these results in order to present a general fairness framework. Note that another possible approach is to contrast the tail asymptotics of $T(x)$ across x, and we refer the reader to the article of Zwart and Boxma in this issue for a survey of such results.

6.1 The experience of large jobs

We now move from a discussion of fairness across all job sizes to a discussion about only *large job sizes*. By focusing on only the behavior of large job sizes, we will be able to study the *distributional behavior* of fairness instead of being limited to studying fairness in expectation. These results will then provide the basis for our generalization of Definition 1 beyond fairness in expectation. Also, a study of large job sizes is of practical importance because it is worries about the performance of large jobs under policies that prioritize small jobs that motivates our study of fairness.

The most natural starting point for generalizing Definition 1 to higher moments is to consider $E[T(x)^i]/x^i$, which corresponds to the study of the distribution of *slowdown*, denoted $S(x) = T(x)/x$. The limiting behavior of slowdown, i.e. $S(x)$ as $x \to \infty$, was first studied by Harchol-Balter, Sigman, & Wierman in the M/GI/1 setting [18]. Following these initial results, a number of other researchers have analyzed the limiting slowdown of nearly all common policies in the GI/GI/1 queue [6, 7, 26]. The results are summarized in the following theorem.

THEOREM 10. *In a GI/GI/1 queue with $E[X^2] < \infty$*

$$\lim_{x\to\infty} S(x) \le_{a.s.} \frac{1}{1-\rho}$$

under all work conserving policies and equality holds for $P \in \{PS, PLCFS, SRPT, FB, SMART\}$ even when $E[X^2] = \infty$. If the policy is also non-preemptive, then $\lim_{x\to\infty} S(x) =_{a.s.} 1$.

Though using $S(x)$ to characterize the distributional behavior of fairness is natural, the fact that $S(x)$ converges almost surely as $x \to \infty$ hints that other metrics with weaker scaling factors may be more appropriate since they will provide more information about the distribution. Specifically, the normalization factor $1/x$ in $S(x)$ hides information of the variability of the response times of large job sizes; thus it is important to consider other scaling factors.

The next natural suggestion for studying the distributional properties of $T(x)$ is to use the *central moments*. However, Wierman & Harchol-Balter [42] illustrate that the central moments provide an appropriate comparison for only the first 3 moments. The observation that the first three central moments are well behaved is important. It suggests that the cumulant moments may provide the correct asymptotic metric.

Cumulants have appeared sporadically in queueing [9, 12, 23], tending to be used in large deviation limits. The cumulant moments of a random variable X, $\kappa_i[X]$ $i = 1, 2, \ldots$, are a descriptive statistic similar to moments and can be found using the logarithm of the moment generating function. Cumulants capture many of the standard descriptive statistics: the first cumulant is the mean; the second is the variance; the third measures the skewness; and the fourth measures the kurtosis. See [21] for tables of the relationships between cumulants, moments, and central moments.

Using the cumulant moments, Wierman & Harchol-Balter [42] obtain a parallel result to Theorem 10 that captures the distributional behavior of $T(x)$ for large job sizes:

THEOREM 11. *In an M/GI/1 queue with $E[X^{i+1}] < \infty$,*

$$\lim_{x\to\infty} \kappa_i[T(x)] \le 1_{[i=1]} + \lambda E[B^i]$$

under all work conserving policies and equality holds for $P \in \{PLCFS, SRPT, FB, SMART\}$ even when $E[X^{i+1}] = \infty$ as long as $E[X^i] < \infty$. If the policy is also non-preemptive, then $\lim_{x\to\infty} \kappa_i[T(x)] = 1_{[i=1]}$.

Theorem 11 can be thought of as a central limit theorem scaling as opposed to the law of large numbers scaling in Theorem 10. It is worth pointing out that though the $1_{[i=1]}$ may appear strange at first, it is a fundamental result of the fact that the first cumulant is shift-equivariant while all others are shift-invariant: letting c be a constant, $\kappa_1[Y+c] = \kappa_1[Y]+c$ but for $i \ge 2$, $\kappa_i[Y+c] = \kappa_i[Y]$.

6.2 A general framework for studying fairness

The results in the previous section provide strong guidance on how to generalize Definition 1 to higher moments. In particular, Theorem 11 motivates the following generalization of the metric $E[T(x)]/x$ and criterion $1/(1-\rho)$:

DEFINITION 6. *Let $0 < \rho < 1$ and $E[X^i] < \infty$ in an M/GI/1. A job size x is treated **fairly** under policy P, service distribution X, and load ρ if*

$$\frac{\kappa_i[T(x)]^P}{x} \le 1_{[i=1]} + \lambda E[B^i].$$

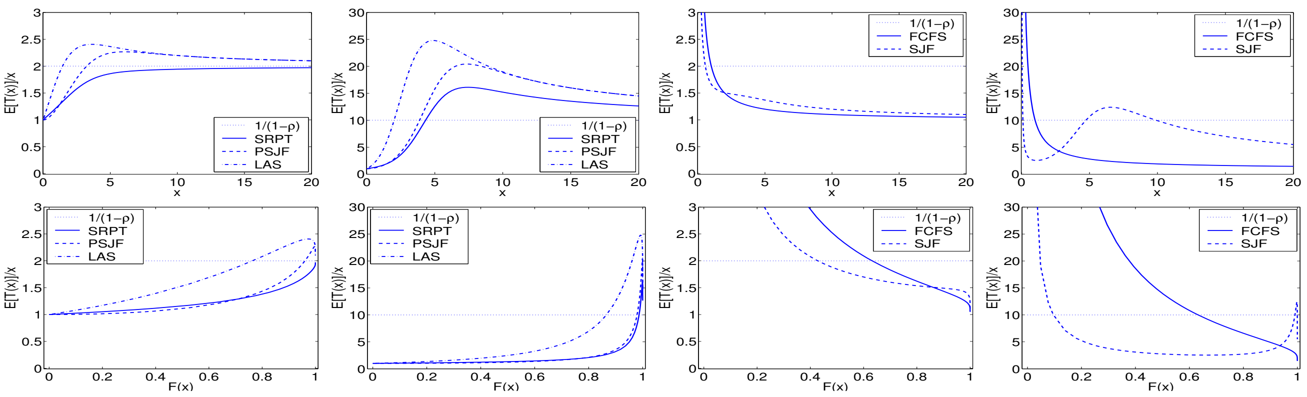
*Otherwise a job size x is treated **unfairly**. A scheduling policy P is **fair** if every job size is treated fairly. Otherwise P is **unfair**.*

Definition 6 was first introduced by Wierman & Harchol-Balter in [42]. However, [42] suggests a more restrictive definition where $E[X^{i+1}] < \infty$. But, the recent results of Brown studying $Var[T(x)]/x = \kappa_2[T(x)]/x$ under SRPT and FB in the setting where $E[X^3]$ is infinite [8] motivate the extension. Note though that we must require $E[X^i] < \infty$ since otherwise $E[B^i] = \infty$.

There are many parallels between Definition 1and Definition 6. The metric and criterion in Definition 1 match the metric and criterion for $i = 1$ in Definition 6: $\kappa_1[T(x)]/x = E[T(x)]/x$ and $1_{[i=1]} + \lambda E[B] = 1 + \rho/(1-\rho) = 1/(1-\rho)$. Further, the motivation for the metric and criterion parallel the motivations for the metric $E[T(x)]/x$ and the criterion $1/(1-\rho)$ in many ways.

In both Definition 1 and Definition 6, the *metric* is motivated by the growth rate of moments of $T(x)$ as $x \to \infty$. Specifically, the metric must scale moments of $T(x)$ appropriately to allow for comparison of moments of $T(x)$ between small and large x. For $E[T(x)]^P$, it is clear that $1/x$ is an appropriate scaling factor because $E[T(x)]^P = \Theta(x)$ under all work conserving scheduling policies. For higher moments of $T(x)$, the correct scaling factor is not obvious; however in Section 6.1 we illustrated that $\kappa_i[T(x)]$ is $\Theta(x)$ for common preemptive policies and $O(x)$ for all work conserving policies. Hence, scaling by $1/x$ makes sense; whereas using a stronger scaling would hide the variability of $T(x)$.

The motivation for the criteria in Definition 6 again parallels that for the criterion in Definition 1; however it is not as cut-and-dry. It is not known whether $1_{[i=1]} + \lambda E[B^i]$ provides a min-max notion of fairness for $i > 1$. Though we conjecture that to be true. But, there are many parallels between $1_{[i=1]} + \lambda E[B^i]$ and $1/(1-\rho)$. For instance, Theorem 11 illustrates that the criterion in Definition 6 also serves as the limit for $\kappa_i[T(x)]/x$ under many common scheduling policies just as $1/(1-\rho)$ does for $E[T(x)]/x$. However, a priori, it is not clear whether this limiting behavior distinguishes between patterns of behaviors with respect



(a) Low load ($\rho = 0.5$) (b) High load ($\rho = 0.9$) (c) Low load ($\rho = 0.5$) (d) High load ($\rho = 0.9$)

Figure 6: The conditional variance of response time is illustrated under a variety of both preemptive and non-preemptive policies. The service distribution is exponential with mean 1. The dotted line shows the criterion for fairness.

to $\kappa_i[T(x)]/x$ in the same way $1/(1-\rho)$ did for $E[T(x)]/x$. But, Wierman & Harchol-Balter [42] proved that in the case of $i = 2$, when $\kappa_2[T(x)]/x = Var[T(x)]/x$, the criterion $\lambda E[B^2]$ does indeed differentiate between contrasting $Var[T(x)]/x$ behaviors and that the behaviors in $Var[T(x)]/x$ parallel those for $E[T(x)]/x$. To illustrate this, we can compare the behavior of $Var[T(x)]/x$ under common policies (shown in Figure 6) with the behavior of $E[T(x)]/x$ (shown in Figure 1). Notice that the Definition 6 distinguishes between between non-monotonic "hump" behaviors – where some mid-range job sizes are treated the most unfairly – and monotonically increasing behaviors in the case of $i = 2$ just as it does in the case of $i = 1$. Similarly, the fairness properties of the scheduling heuristics and techniques studied in Section 4 are parallel for $E[T(x)]/x$ and $Var[T(x)]/x$ [42]. An important open question that remains though is to show that $\kappa_i[T(x)]$ exhibits parallel behavior for $i > 2$.

7. OTHER FAIRNESS METRICS

Up to this point, we have focused on a specific type of fairness, that of guaranteeing equitable response times to all job sizes. This focus is motivated by fears about recent computer system designs that propose policies which give priority to small jobs at the expense of large jobs. However, as we discussed earlier, different settings require different notions of fairness. As a result, following the introduction of Definition 1, many other metrics have emerged for evaluating the fairness of scheduling policies.

In particular, Definition 1 only captures the notion of *proportional fairness*, i.e. the idea that it is fair for jobs to receive response times proportional to their service times. Resultantly, it is unfair to force a small job to queue behind a large job because the response time of the small job will become unfairly large. However, proportional fairness is not always appropriate because if a large job has been waiting in the queue for a long time it is in some sense unfair for a small job that just arrived to the queue to jump in front of the large job, e.g. the queue at a ticket box office. In fact, in many settings `FCFS` is the most fair policy because it respects the seniority of customers – this is the idea of *temporal fairness*.

Recently, a number of measures have been suggested to capture the notion of temporal fairness, and in many cases balance it against proportional fairness. Unfortunately, we do not have the space in this short article to provide a detailed survey of these measures, but three of the most developed of these measures are: Order Fairness [1], Resource Allocation Queueing Fairness Measure (RAQFM) [32], and Discrimination Frequency (DF) [35]. Each of these measures captures a different notion of what is "fair."

Order Fairness, introduced by Avi-Itzhak & Levy [1], was developed using four simple axioms to formalize the concept of temporal fairness in a G/D/1 queue. Interestingly, for non-preemptive policies, Avi-Itzak & Levy determine that the variance of delay satisfies these four axioms and is thus a useful measure of temporal fairness.

Following the introduction of the Order Fairness measure, Avi-Itzhak & Levy in combination with Raz present a fairness measure that balances both proportional and temporal fairness called RAQFM [32]. The idea behind RAQFM is that at every point in time, every job in the system deserves an equal service rate. Thus, the fairness of a policy is determined by looking at the variation from this service rate that jobs experience. Avi-Itzhak, Levy, & Raz, have a number of papers illustrating the properties of this interesting measure [31, 2], but unfortunately, the fairness of many practically relevant policies, e.g. `SRPT` and `FB`, has proven difficult to analyze using RAQFM.

Shortly after the introduction of RAQFM, Sandmann introduced another fairness measure called Discrimination Frequency (DF) [35]. Unlike RAQFM, which implicitly balanced the notions of proportional fairness and temporal fairness, DF is defined as an explicit combination of two measures, one for proportional fairness and one for temporal fairness, each of which is easily analyzed under all common scheduling policies. A key benefit of this approach is that DF allows the measure to be tuned depending on the application needs. The two measures DF combines are (i) the number of jobs that arrived after and completed before the tagged job and (ii) the number of jobs with remaining size larger than the tagged job (upon arrival) that complete before the tagged job.

Clearly, each of these three fairness measures captures a different aspect of what is meant by the term "fair," and resultantly none is suitable across all applications. However, understanding the performance of a scheduling policy under all of these measures provides significant insight into the behavior of the policy.

8. CONCLUSION

The goal of this article is to provide a short survey of recent work studying the question: "are policies that prioritize small jobs unfair to large jobs?" Much work has appeared on this topic since 2001, including a number of surprising results, e.g. that policies such as SRPT and FB can be fair to all job sizes in many practical settings. However, it is clear from this survey that many interesting and important questions remain.

Within the context of Definition 6, it would be very interesting to determine whether the behavior of policies (and scheduling heuristics) with respect to higher moments matches what has been observed for the first two moments. Further, it would be interesting to determine whether the criterion in Definition 6 provides a min-max notion of fairness beyond the first moment. Additionally, obtaining results for FSP in the M/GI/1 setting is of great interest. Finally, beyond Definition 6, there are an abundance of other important questions to address: from contrasting recently presented notions of fairness, to finding policies that perform well across all fairness measures while still providing near optimal $E[T]$, to extending definitions of fairness beyond the single server model.

Though many open questions remain, the study of fairness has already begun to have a dramatic effect on the design of computer systems. The counter-intuitive fact that it is possible to give priority to small job sizes without hurting the performance of large job sizes has led to the gradual acceptance of designs using policies such as SRPT, PSJF, and FB in a variety of applications, including web servers [17, 29], operating systems [10], routers [27, 28], wireless networks [19], transport protocols [45], and beyond.

9. REFERENCES

[1] B. Avi-Itzhak and H. Levy. On measuring fairness in queues. *Adv. of Appl. Prob.*, 36(3):919–936, 2004.

[2] B. Avi-Itzhak, H. Levy, and D. Raz. Quantifying fairness in queueing systems: Principles, approaches and applicability. *Prob. in the Eng. and Info. Sciences*, in press.

[3] Baily, Foster, Hoang, Jette, Klingner, Kramer, Macaluso, Messina, Nielsen, Reed, Rudolph, Smith, Tomkins, Towns, and Vildibill. Valuation of ultra-scale computing systems. White Paper, 1999.

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

[5] M. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continous job streams. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[6] S. Borst, O. Boxma, R. Núñez-Queija, and B. Zwart. The impact of the service discipline on delay asymptotics. *Performance Evaluation*, 54:175–206, 2003.

[7] S. Borst, R. Núñez-Queija, and B. Zwart. Sojourn time asymptotics in processor-sharing queues. *Queueing Sys.*, 53(1-2), 2006.

[8] P. Brown. Comparing FB and PS scheduling policies. *Perf. Eval. Rev.*, 34(3), 2006.

[9] N. Duffield, W. Massey, and W. Whitt. A nonstationary offered-load model for packet networks. *Telecommunication Systems*, 13:271–296, 2001.

[10] H. Feng, V. Misra, and D. Rubenstein. PBS: a unified priority-based cpu scheduler. In *Proc. of ACM Sigmetrics*, 2007.

[11] E. Friedman and S. Henderson. Fairness and efficiency in web server protocols. In *Proc. of ACM Sigmetrics*, 2003.

[12] G.L.Choudhury and W. Whitt. Heavy-traffic asymptotic expansions for the asymptotic decay rates in the BMAP/G/1 queue. *Stochastic Models*, 10:453–498, 1994.

[13] M. Gong and C. Williamson. Quantifying the properties of SRPT scheduling. In *IEEE/ACM Symposium on Mod., Anal., and Sim. of Comp. and Telecomm. Sys. (MASCOTS)*, 2003.

[14] M. Gong and C. Williamson. Simulation evaluation of hybrid SRPT scheduling policies. In *Proc of IEEE MASCOTS*, 2004.

[15] M. Gong and C. Williamson. Revisiting unfairness in web server scheduling. *Computer Networks*, 50(13):2183–2203, 2006.

[16] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *Transactions on Computer Systems*, 15(3), 1997.

[17] M. Harchol-Balter, B. Schroeder, M. Agrawal, and N. Bansal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[18] M. Harchol-Balter, K. Sigman, and A. Wierman. Asymptotic convergence of scheduling policies with respect to slowdown. *Performance Evaluation*, 49(1-4):241–256, 2002.

[19] M. Hu, J. Zhang, and J. Sadowsky. A size-aided opportunistic scheduling scheme in wireless networks. In *Globecom*, 2003.

[20] F. Kelly. *Reversibility and Stochastic Networks*. John Wiley & Sons, 1979.

[21] M. Kendall. *The Advanced Theory of Statistics*. Griffin, London, 1945.

[22] A. A. Kherani and R. Núñez-Queija. TCP as an implementation of age-based scheduling: fairness and performancs. In *IEEE Infocom*, 2006.

[23] T. Matis and R. Feldman. Using cumulant functions in queueing theory. *Queueing Sys.*, 40:341–353, 2002.

[24] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. Gehrke. Online scheduling to minimize average stretch. In *Proc. of Found. of Comp. Sci.*, 1999.

[25] M. Nuyens and A. Wierman. The foreground-background queue: A survey. *Under submission*, 2007.

[26] M. Nuyens, A. Wierman, and B. Zwart. Preventing large sojourn times using SMART scheduling. *Operations Research*, in press.

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

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

[29] M. Rawat and A. Kshemkalyani. SWIFT: Scheduling in web servers for fast response time. In *Symp. on Net. Comp. and App.*, 2003.

[30] J. Rawls. *A theory of social justice*. Harvard University Press, 1971.

[31] D. Raz, B. Avi-Itzhak, and H. Levy. Fairness considerations in multi-server and multi-queue systems. In *Valuetools*, 2006.

[32] D. Raz, H. Levy, and B. Avi-Itzhak. A resource-allocation queueing fairness measure. In *Proc. of ACM Sigmetrics-Performance*, 2004.

[33] R. Rhodes, M. P. Battin, and A. Silvers. *Medicine and social justice*. Oxford University Press, 2002.

[34] R. Righter, J. Shanthikumar, and G. Yamazaki. On external service disciplines in single stage queueing systems. *J. of Applied Probability*, 27:409–416, 1990.

[35] W. Sandmann. A discrimination frequency based queueing fairness measure with regard to job seniority and service requirement. In *Proc. of Euro NGI Conf. on Next Generation Int. Nets*, 2005.

[36] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.

[37] L. E. Schrage and L. W. Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*, 14:670–684, 1966.

[38] A. Silberschatz and P. Galvin. *Operating System Concepts, 5th Edition*. John Wiley & Sons, 1998.

[39] W. Stallings. *Operating Systems, 2nd Edition*. Prentice Hall, 1995.

[40] A. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 1992.

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

[42] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to higher moments of response time. In *Proc. of ACM Sigmetrics*, 2005.

[43] A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *Proc. of ACM Sigmetrics*, 2005.

[44] C. Yang, A. Wierman, S. Shakkottai, and M. Harchol-Balter. Tail asymptotics for policies favoring short jobs in a many-flows regime. In *Proc. of ACM Sigmetrics*, 2006.

[45] S. Yang and G. de Veciana. Enhancing both network and user performance for networks supporting best effort traffic. *Trans. on Networking*, 12(2):349–360, 2004.