Revisiting Unfairness in Web Server Scheduling

Mingwei Gong Carey Williamson
Department of Computer Science
University of Calgary
Email: {gongm,carey}@cpsc.ucalgary.ca

August 24, 2005

Abstract

This paper uses trace-driven simulation to study the unfairness properties of Web server scheduling strategies, such as Processor Sharing (PS) and Shortest Remaining Processing Time (SRPT). We use a general-purpose probe-based sampling approach to estimate the mean and variance of the job response time for different job sizes, for arbitrary arrival processes and service time distributions. The results illustrate two different aspects of unfairness called *endogenous unfairness* and *exogenous unfairness*. We quantify each, focusing on the mean and variance of slowdown conditioned on job size, for a range of system loads. Our work confirms recent theoretical results regarding the asymptotic convergence of scheduling policies with respect to slowdown, and illustrates typical performance results for a practical range of job sizes in an empirical workload. Finally, we show the sensitivities of SRPT and PS scheduling to selected characteristics of the arrival process and job size distribution.

Keywords: Web Server Performance, Scheduling, Fairness, Simulation, Performance Analysis

1 Introduction

The Shortest Remaining Processing Time (SRPT) scheduling policy has received increasing attention in the research literature recently, primarily in the context of request scheduling at Web servers [4, 7, 15]. The SRPT policy selects for service the pending job in the system with the least remaining service time. The policy is preemptive, so that if a new job arrives into the system with a smaller service time than the job currently in service, the scheduler switches immediately to service the newly arriving job. The SRPT policy is provably optimal: it guarantees the lowest mean response time for the system as a whole [28, 32].

The primary concern with SRPT is unfairness: a large job in the system may *starve* if the continuous arrival of small(er) jobs preempts it from service. Prior results in the literature clearly establish the advantages of SRPT over a conventional scheduling policy such as Processor

Sharing (PS), particularly for small jobs. Small jobs are serviced much more quickly under SRPT scheduling than under PS scheduling. However, jobs at the upper end of the job size distribution may experience worse performance under SRPT than under PS. Bansal and Harchol-Balter illustrate this clearly in several of their papers [4, 15].

Harchol-Balter *et al.* [16] have recently established asymptotic bounds on the slowdown performance for the largest jobs under SRPT (or any other) scheduling policy. In particular, their results show that the slowdown metric (defined as the job response time divided by the job size) asymptotically converges to the *same* value for *any* scheduling policy. In other words, for the largest of jobs, SRPT is no worse than PS. In addition, they prove that for *sufficiently large* jobs, the slowdown performance under SRPT is only marginally worse than under PS, by at most a factor of $1 + \epsilon$, for small $\epsilon > 0$. In our paper, we use the term "crossover region" to refer to the range of job sizes for which SRPT provides worse slowdown than PS.

There are two main objectives of this paper. The first objective is to study the relationship between prior theoretical work and the performance for typical job sizes at an SRPT Web server, based on empirical data. For example, it is not clear what "sufficiently large" [16] means in practice. As another example, is the crossover region observable in practice, and if so, for what range of job sizes does it occur? The second objective is to study the effects of the request arrival process and the job size distribution on scheduling policy performance. Synthetic traces are used to study these impacts.

Our work is carried out using trace-driven simulation. We use a probe-based sampling methodology [10, 11] to evaluate job slowdown for SRPT and PS scheduling policies in a simulated Web server system, with an empirical Web request stream for static Web content from the 1998 World Cup Web site [1]. The sampling methodology provides a robust means of estimating the mean and variance of job slowdown as a function of job size and system load, enabling a methodical study of the performance differences between SRPT and other scheduling policies.

There are three main contributions in this paper. First, our results show that there are two types of unfairness in a Web server scheduling system: *endogenous unfairness* that a job can suffer because of its own size, and *exogenous unfairness* that a job can suffer because of the state of the Web server (i.e., other jobs in the system) when it arrives. By revisiting the notion of unfairness more carefully, and quantifying these effects separately, we provide new insights into the differences between the "unfair" SRPT policy and the "fair" PS policy. Second, we confirm prior theoretical results in the literature, quantifying their presence in an empirical workload. Third, we illustrate the impacts of the request arrival process and the service time distribution on the performance of Web server scheduling policies.

The rest of the paper is organized as follows. Section 2 provides background information on Web server performance and SRPT scheduling. Section 3 motivates and explains our probe-based sampling methodology. Section 4 describes the experimental methodology for our simulation study. Section 5 focuses on the empirical trace, and Section 6 uses synthetic traces that vary the request arrival process and the job size distribution. Finally, Section 7 concludes the paper.

2 Background and Related Work

2.1 Web Server Performance

Web server performance is a popular theme in the recent research literature [1, 16, 22]. The user-perceived performance for Web browsing depends on many factors, including server load, network load, and the protocols used for client-server interaction. In this paper, we focus on one aspect of Web server configuration, namely the scheduling policy for servicing HTTP requests.

The scheduling policy used at the Web server determines the relative order of service for incoming client requests. The simplest scheduling policy, assuming a single-process Web server, is First-Come-First-Serve (FCFS): requests are served serially in the order of their arrival. In practice, most Web servers use multi-process or multi-threaded designs. With this approach, many requests (typically hundreds to thousands) can be in progress at a time, each sharing the available CPU, I/O, and network resources. This approach is commonly approximated with the Processor Sharing (PS) scheduling discipline: if there are N requests pending in the system, then each request receives service at a rate 1/N of the maximal rate. This approach is deemed "fair" because it shares resources equally amongst contending requests.

The Shortest Remaining Processing Time (SRPT) first policy is a preemptive scheduling policy that optimizes mean job response time. Using advance knowledge of job service time requirements, the SRPT policy always selects for service the job that has the least remaining service time. With this approach, the mean waiting time and the mean job response time are minimized [28, 32].

2.2 Related Work

The early theoretical work on SRPT scheduling in queueing systems was done over 30 years ago [28, 29]. This topic has seen renewed activity in the last 10 years as well [24, 27, 30].

The investigation of SRPT scheduling for Web servers began in the late 1990's [7]. There are solid theoretical underpinnings for the idea [4, 16], as well as a prototype implementation of SRPT scheduling in the Apache Web server [15]. Experimental results confirm many of the performance advantages of SRPT scheduling established in the theoretical work. The SRPT policy is also effective at combatting system overload, since it minimizes the number of jobs that are starved [31].

Despite this solid theoretical and experimental work in the literature, concerns remain about the unfairness of SRPT scheduling, with respect to starvation and unbounded slowdown for the largest jobs. The SRPT policy is not yet widely deployed in Internet Web servers, in part because there is incomplete understanding of its behaviour for empirical Web workloads. It is on this front that our paper makes one of its main contributions.

Recent research has spawned several other scheduling policies that attempt to capture the advantages of SRPT, but improve upon its fairness properties. Examples of these policies include the Fair Sojourn Protocol (FSP) [9], Least Attained Service (LAS) [25], Resource Allocation Queueing Fairness Measure (RAQFM) [26], and K-SRPT [12]. The FSP protocol combines aspects of PS and SRPT. It selects for service the pending job that would complete the earliest

under PS scheduling, and then devotes full service to this job until the next arrival or departure event (which may change the system state). The LAS policy approximates the effectiveness of SRPT, without the need to know job sizes in advance. In fact, Lu *et al.* [19] show that the performance of SRPT deteriorates a lot if it does not have precise job size information. The idea behind RAQFM is to balance fairness between the service requirement (size) and seniority of a job. The K-SRPT policy generalizes SRPT to have up to K jobs simultaneously in service (on a PS basis), for the K jobs with the smallest remaining service times.

All of these policies in the literature demonstrate the advantages of biasing size-based scheduling in favour of small jobs. The policies differ in how well they improve upon the unfairness of SRPT. In our paper, we focus primarily on PS and SRPT, which arguably demarcate the two endpoints for this broad spectrum of scheduling policies.

3 Sampling Methodology

This section motivates and explains our probe-based sampling methodology for assessing the unfairness of SRPT scheduling. Section 3.1 presents a simple example to provide some insight into the dynamic behaviour of scheduling policies. Section 3.2 introduces our sampling method.

3.1 Preliminaries: Understanding SRPT Scheduling

Figure 1 provides a simple example of a request stream constituting a Web server workload. This example has 20 requests, in order of their arrival. The two-column format shows the timestamp (in seconds) and the response (job) size (in bytes) for each request. This is the trace format assumed for Web server workloads throughout the paper.

Figure 2 provides a graphical illustration of the busy period structure for FCFS, PS, SRPT, and LRPT (Longest Remaining Processing Time) scheduling at a Web server processing this input request stream. We assume that the Web server is idle when the first request arrives.

Figure 2(a) shows the instantaneous number of jobs in the system for the FCFS policy on this workload, as a function of time. The vertical upward steps represent job arrivals, and the vertical downward steps represent job departures. Figure 2(b) shows the corresponding number of bytes in the system for the FCFS scheduler. The vertical upward spikes represent job arrivals, which can be of arbitrary size. The downward slope represents the byte service rate when the server is busy. Whenever this downward slope meets the horizontal axis, the current busy period ends, and the server remains idle until the next job arrival.

Figure 2(c) shows the instantaneous number of jobs in the system for the PS policy on the same workload, and Figure 2(d) shows the corresponding number of bytes. Figures 2(e) and (f) show the results for the SRPT policy, while Figures 2(g) and (h) show the results for LRPT.

Three observations are evident from Figure 2. First, the profiles for "byte backlog" are identical for each of the scheduling policies considered (though the job departure points may differ). This obvious property holds for *any* work-conserving scheduling policy on this workload, assuming the same job arrival times, the same job sizes, and the same byte service rate. Second, the start and end times of the busy periods are the same for each policy. This follows directly from

TIMESTAMP	SIZE
0.000000	3,038
0.000315	949
0.001048	2,240
0.004766	2,051
0.005642	366
0.005872	201
0.006380	298
0.006742	1,272
0.007271	597
0.008008	283
0.008653	482
0.010165	852
0.010911	929
0.013306	191
0.013969	1,005
0.016681	322
0.016961	1,420
0.017391	191
0.018563	3,867
0.018783	914

Figure 1: Simple Example of Web Server Workload Request Stream

the first observation. What this means is that the number of busy periods, as well as the mean and variance of the busy period duration, are *invariant* across (work-conserving) scheduling policies. This invariant propery provides a useful validation check on the simulation implementations of different scheduling policies. Third, and most important, the distribution of the number of jobs in the system is *different* for each of the policies considered. For the small sample workload considered here, the SRPT policy never has more than 3 jobs simultaneously in the system, while the PS policy has up to 5 jobs in the system, and LRPT has up to 11 jobs in the system at a time.

The tradeoff between PS and SRPT scheduling is now more evident. With PS scheduling, an arriving job receives immediate service, but its service rate may be low because of the (larger) number of jobs in the system. With SRPT scheduling, a job either receives immediate service at the maximal rate (if it has the least remaining service time requirement), or receives no service while it waits (if it is not). The probability of immediate service depends in part on the number of jobs in the system, but mostly on the relative sizes of the competing jobs. Simple intuition suggests that the fewer competing jobs in the system, the sooner service will be received, but this is not necessarily true for SRPT.

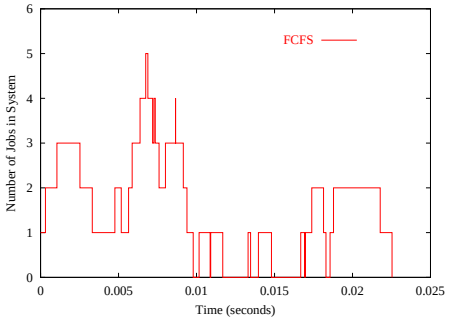
The difference in "jobs in the system" is precisely the property that we focus on in the experiments to follow. In fact, our probe-based sampling methodology is designed to estimate the impact of this property on the response time of a job, from the job's perspective. A description of our sampling algorithm follows.

3.2 Probe-based Sampling Algorithm

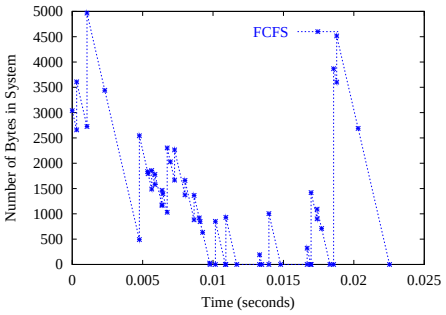
Figure 3 provides a high-level algorithmic description of our sampling methodology for quantifying the unfairness properties of SRPT scheduling compared to PS and other scheduling policies. The sampling methodology is probe-based, and relies on the PASTA principle: Poisson Arrivals See Time Averages.

The algorithm works as follows. Given a pre-defined system load (i.e., Web request arrival stream) and a scheduling policy at the Web server, a single *probe job* is inserted at random into the request arrival stream, and the Web server is simulated using the modified request stream to determine the response time for the probe job. By repeating the experiment N times (e.g., $N = 3000$, in our experiments) with random placement (according to the PASTA principle) of the single probe job in the request stream, we obtain samples of the response time distribution for a job of that size. By repeating the experiment with different probe job sizes, we can assess the unfairness properties of a specific scheduling policy. We can also vary the system load to determine the load level at which unfairness is most pronounced, for a particular job size.

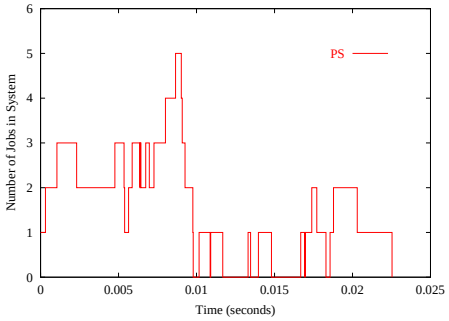
The straightforward naive implementation of the algorithm in Figure 3 would be very compute-intensive, since it requires many executions of the Web server simulator, each with a slightly modified version of the request stream. In practice, there are several optimizations to expedite the simulations. For example, there is no need to re-simulate all the busy periods that complete prior to the arrival of the probe job. Rather, it suffices to simulate (in its entirety) the busy period in which the probe job arrives and completes. Similarly, there is no need to simulate all busy periods that follow the busy period in which the probe job completes. With these optimiza-



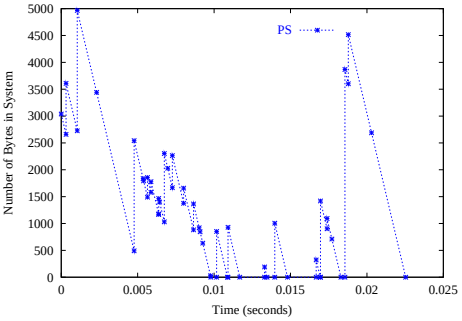
(a) Jobs in System (FCFS)



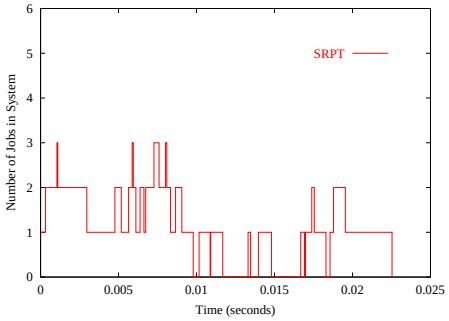
(b) Backlog in Bytes (FCFS)



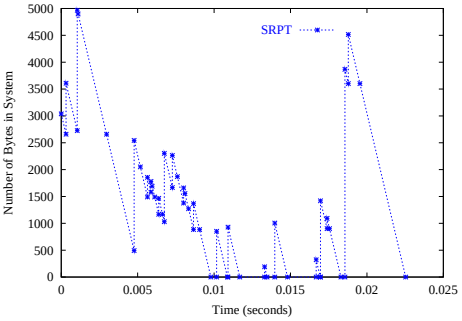
(c) Jobs in System (PS)



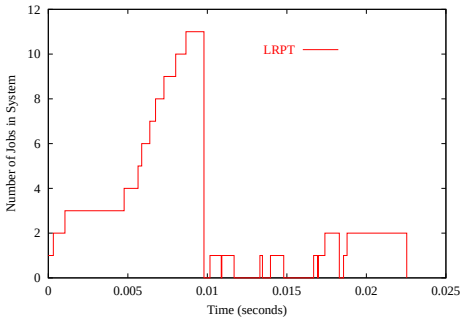
(d) Backlog in Bytes (PS)



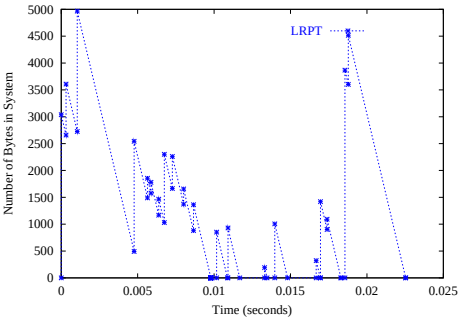
(e) Jobs in System (SRPT)



(f) Backlog in Bytes (SRPT)



(g) Jobs in System (LRPT)



(h) Backlog in Bytes (LRPT)

Figure 2: Simulation Results Illustrating Busy Period Structure for FCFS, PS, SRPT, and LRPT Scheduling on the Example Workload

```
For scheduling algorithm S = ( PS, SRPT, LRPT, FCFS, FSP, ... ) do
  For background load level U = ( 0.50, 0.80, 0.95 ) do
    For probe job size J = ( 1 KB, 10 KB, 100 KB, 1 MB, 10 MB ) do
      For trial i = ( 1, 2, 3, ... N ) do
        Insert probe job at randomly chosen point in original request stream
        Simulate Web server scheduling policy on modified request stream
        Compute and record response time and slowdown for probe job
      end for i

      Plot marginal distribution of slowdown for this J, U, S combination
    end for J
  end for U
end for S
```

Figure 3: Algorithmic Overview of Sampling Methodology Using Probe Jobs

tions, the algorithm in Figure 3 is computationally feasible, though clearly the greatest overhead occurs for high system loads, since idle periods are few and far between.

Our current implementation of the algorithm uses a checkpointing technique so that all job probes can be simulated using a single pass through the workload stream. Additional optimizations could exploit parallelism in simulating probe jobs that affect disjoint busy periods; we have not yet investigated this technique.

4 Simulation Methodology

4.1 Simulation Model

Trace-driven simulation is used to evaluate the performance of different scheduling policies on a simulated Web server. The input trace to the simulator follows the format introduced in Figure 1, namely a two-column file containing request arrival time and response size in bytes.

The Web server model in the simulation is simple. A configuration parameter specifies the service rate for the server, in bytes per second. Outgoing network bandwidth is assumed to be the bottleneck. This assumption is consistent with prior work on SRPT scheduling [15, 31]. A second configuration parameter specifies the scheduling policy to be used. Currently, our simulator supports FCFS, PS, SRPT, LRPT, and FSP. A request that arrives to an idle server begins service immediately at the specified byte service rate. A request that arrives to a busy server either waits its turn (FCFS policy, and possibly SRPT and LRPT depending on job sizes), or begins service immediately (PS policy, and possibly SRPT and LRPT depending on job sizes). The PS policy adjusts the per-job service rate dynamically at the time of arrival and departure events based on the number of jobs in the system. The SRPT and LRPT policies dynamically

Table 1: Characteristics of Empirical Web Server Workload Used (World Cup 1998)

Item	Value
Trace Name	wc_day66_6.gz
Trace Date	June 30, 1998
Trace Duration	861 sec
Total Requests	1,000,000
Unique Documents	5,549
Total Transferred Bytes	3.3 GB
Smallest Transfer Size (bytes)	4
Largest Transfer Size (bytes)	2,891,887
Median Transfer Size (bytes)	889
Mean Transfer Size (bytes)	3,498
Standard Deviation (bytes)	18,815

choose the next job to service, based on remaining bytes, preempting as necessary.

Instrumentation in the simulator records job arrivals, job departures, and the byte backlog in the system at arrival and departure events. The simulation also records information about busy periods, idle periods, and the number of jobs and bytes in the system during busy periods.

4.2 Web Server Workload Trace

The primary Web server workload used in our experiments is an empirical trace from the 1998 World Cup Web site [1, 17]. The same trace is also used by Schroeder *et al.* [31]. As in prior SRPT work, we assume that all requests are for static Web content. The trace selected has 1 million requests, representing an elapsed time duration of just over 14 minutes. The average request arrival rate is 1160 requests per second. The largest 1% of the transfers account for 20% of the bytes transferred. Additional trace characteristics are discussed in [12].

Table 1 provides further information about the trace. In our experiments, we augment the one-second resolution timestamps in the trace to distribute requests uniformly at random across each one-second interval, while preserving the order of request arrivals. The modified timestamps are used to determine the request arrival process.

4.3 Experimental Design

The experiments use a multi-factor experimental design. The primary factors of interest are scheduling policy, job size, and system load. The scheduling policies considered are PS and SRPT. System load is controlled by setting the byte service rate (i.e., network link capacity) for the server. We consider probe job sizes ranging from 100 bytes to 10 MB, since this (arguably) spans the most relevant range for typical Web object sizes. The largest probe job size considered (10 MB) represents a "small" perturbation to the total load (i.e., it increases the total transferred

Table 2: Experimental Factors and Levels for Web Server Scheduling Experiments

Factor	Levels
System Load (U)	0.50, 0.80, 0.95
Probe Job Size (J)	100 bytes, 1 KB, 10 KB, 100 KB, 1 MB, 10 MB
Scheduling Policy (S)	FCFS, PS, SRPT, LRPT

bytes by less than 0.3%). However, it is larger than any other job size in the empirical workload, and depending on the probe arrival time, could extend the simulated completion time slightly.

Table 2 summarizes the factors and levels used in our experiments. For space reasons, only a subset of the experiments is reported in this paper. Additional results are available in [10].

4.4 Performance Metrics

The simulation experiments use the following performance metrics:

- *Number of jobs in the system*: This metric is used in time series plots and in frequency histogram plots to illustrate the behaviours of different scheduling policies.

- *Number of bytes in the system*: This metric is used to validate the correct operation of different scheduling policies, as discussed in Section 3.1.

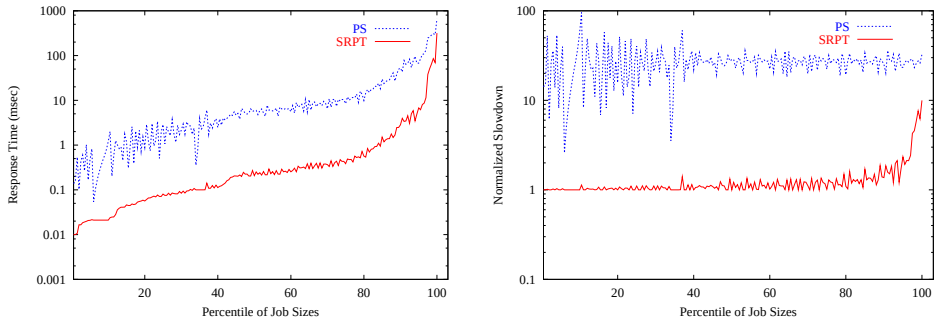
- *Response time*: This metric is used to measure the response time for the probe job in our sampling methodology. Response time is defined as the elapsed time from when the request first arrives in the system until it departs from the system.

- *Slowdown*: The slowdown metric is used to measure the performance for the probe job in our sampling methodology. We define *Slowdown* as the *response time* of a job divided by the *ideal response time* if it were the sole job in the system. This metric is often referred to as normalized response time, inflation factor, or stretch factor in the literature [21]. Lower values of the slowdown metric represent better system performance.

  The slowdown metric is used in time series plots and in frequency histogram plots to illustrate the behavioural properties of different scheduling policies. The mean slowdown, where used, is the average slowdown computed across all samples.

- *Coefficient of Variation (CoV) of slowdown.*

  There are several proposed definitions for "fairness" in Web server scheduling [3, 4, 26, 33]. We focus on an application-layer measure of job-level fairness, rather than a network-layer measure of flow-level fairness (such as Max-Min fairness [14]). In particular, we use the CoV of slowdown to measure the degree of unfairness. In a perfectly fair environment, the slowdown of different requests should be the same, so the CoV is zero. The larger the CoV value is, the greater the unfairness.



(a) Response Time (b) Normalized Slowdown

Figure 4: Simulation Validation Results for PS and SRPT Scheduling Policies (U = 0.95)

4.5 Validation

Significant effort focused on verification and validation of the results reported by our simulator. This section briefly describes several of these steps.

The first validation step involved testing our simulator on short traces such as that in Figure 1, for which results could be verified by hand. We verified that the busy period behaviour was correct, and that the byte backlog process was consistent for all scheduling policies considered.

The second validation step involved testing our sampling approach to ensure that it followed the PASTA principle. Our probe generation approach passed these tests, indicating that the system state is sampled in a Poisson fashion.

The third validation step compared slowdown results reported by our simulator to published results in the literature for the SRPT and PS policies (albeit for different traces). An example of these simulation results is provided in Figure 4, for our workload trace. Figure 4(a) shows job response time, while Figure 4(b) shows the normalized slowdown metric, both plotted versus the percentile of job sizes in the job size distribution, following the format used in [4]. Our simulation results are qualitatively consistent with those reported in [4], providing further confidence in the results reported by our simulator.

One additional validation test studied the number of busy periods, and how the number of busy periods is affected by the insertion of the probe job. Three cases are possible:

- *The probe job can increase the number of busy periods, by one.* This case occurs if the probe job arrives in an idle period, and is completely served within that (formerly) idle period. The probability of this occurrence depends on the probe job size and the proportion of time that the server is idle. Tests with infinitesimal (1 byte) probe jobs produced results consistent with the level of system load.

- *The probe job can leave the number of busy periods unchanged.* This case occurs if the probe job arrives in (or just slightly before) an existing busy period, and is serviced to completion in that (slightly extended) busy period, *without* merging with the following busy period.

- *The probe job can reduce the number of busy periods, by one or more.* This case occurs if

the addition of the probe job causes two or more busy periods to coalesce. The coalescence case is common for large probe job sizes, especially at moderate and high loads.

Analysis of the busy period behaviour in our experiments was consistent with the explanations provided here. Figure 5 provides an example of the busy period analysis from 300 random probes, for different probe job sizes at 95% system load. For a 1 KB probe job size, the insertion of the probe job adds at most one busy period to the initial total of 49,714 busy periods, and removes at most two. For a 10 KB probe job size, up to 8 busy periods coalesce. For a 100 KB probe job size, about 30 busy periods coalesce on average, while for 1 MB probe jobs, about 300 busy periods coalesce on average.

This suite of validation tests confirms that our Web server simulator is working correctly. The validation process establishes confidence in the simulation results, which we present in the following sections.

5 Simulation Results: Empirical Trace

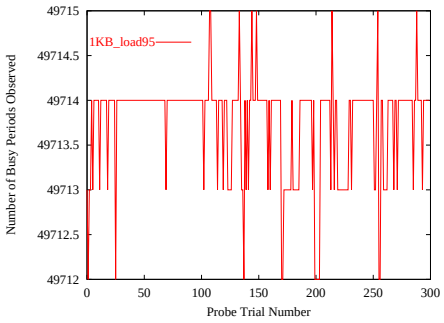
5.1 General Observations

Figure 6 provides a high-level view of the key differences between the PS and SRPT policies. These graphs show short (60 second) time series plots for the number of jobs simultaneously in the system for PS and SRPT scheduling policies on the empirical Web server workload trace, as well as the marginal distribution (frequency histogram) of the number of jobs in the system, based on the full empirical trace. The results are illustrated for three different levels of system load (50%, 80%, and 95%), from top to bottom in Figure 6. Note that the vertical scales of the graphs are different for each load level considered.

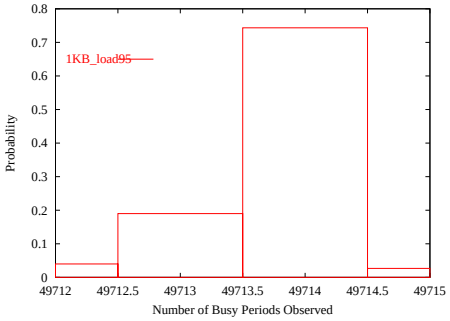
The top row of graphs in Figure 6 shows the results for 50% load. The time series plots show the number of jobs in the system for each scheduling policy: PS in Figure 6(a), and SRPT in Figure 6(b). Figure 6(c) shows the resulting marginal distributions. In this graph, there is little difference between the marginal distributions for PS and SRPT. Both plots start at 0.5, since the server is idle half the time (by definition of the system load), and tail off relatively quickly after that. At this modest level of load, it is rare to have more than 10 jobs in the system at a time, with either policy.

The second row of graphs in Figure 6 shows the results for 80% load. Again, two time series plots are shown (PS in Figure 6(d), and SRPT in Figure 6(e)), with the marginal distribution results in Figure 6(f). At 80% load, the differences between policies are more apparent. While both plots start at 0.20 (corresponding to 80% load), the marginal distribution for the SRPT policy is very "tight", while that for the PS policy has a much longer tail. The means of the distributions differ significantly.

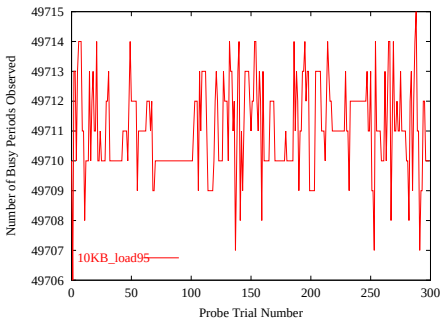
These differences are even more apparent in the third row of graphs in Figure 6, which shows results for 95% load. In Figure 6(i), the marginal distribution for SRPT is very tight; there are never more than 30 jobs in the system at a time for this workload. The marginal distribution for the PS policy shows a much longer tail, with up to 180 jobs in the system at a time.



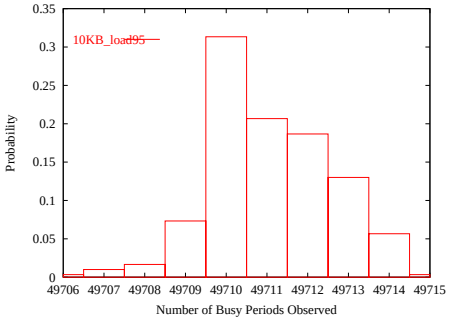
(a) J = 1 KB



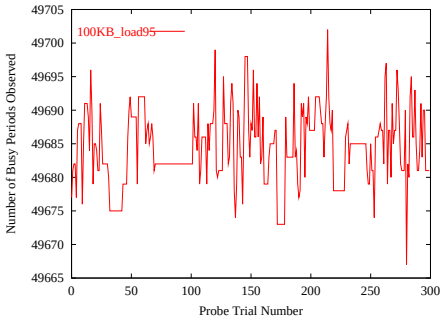
(b) Marginal Dist.



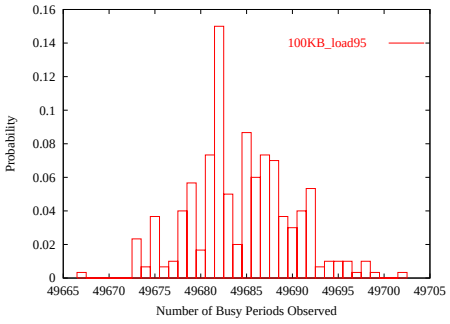
(c) J = 10 KB



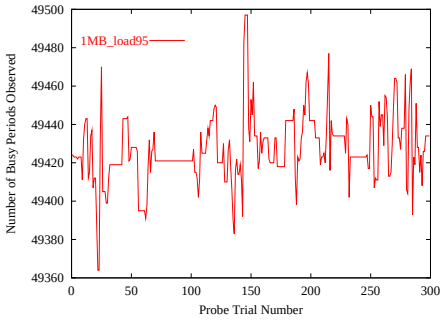
(d) Marginal Dist.



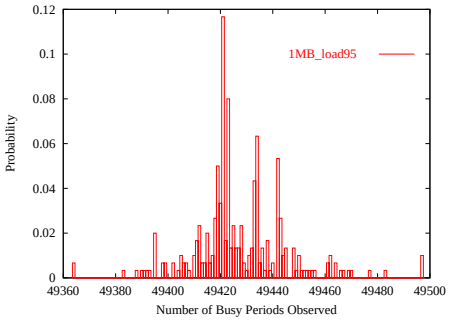
(e) J = 100 KB



(f) Marginal Dist.

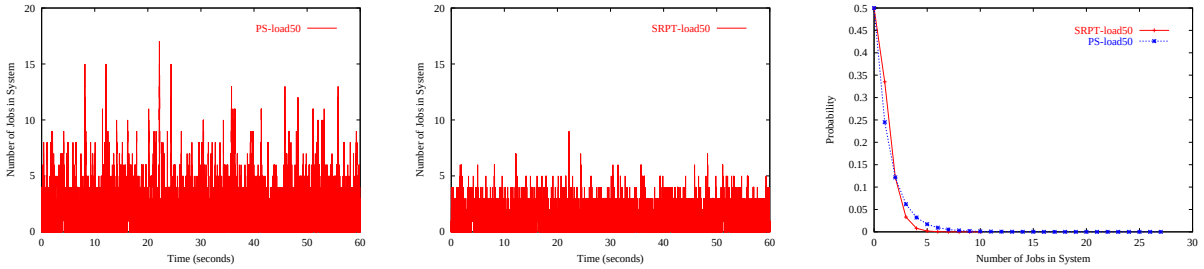


(g) J = 1 MB

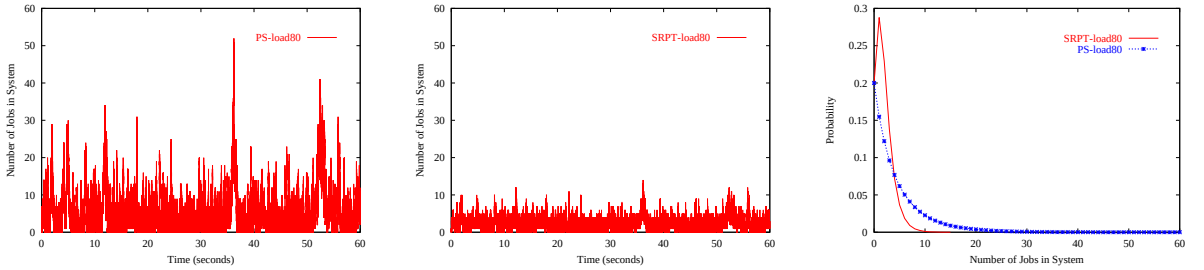


(h) Marginal Dist.

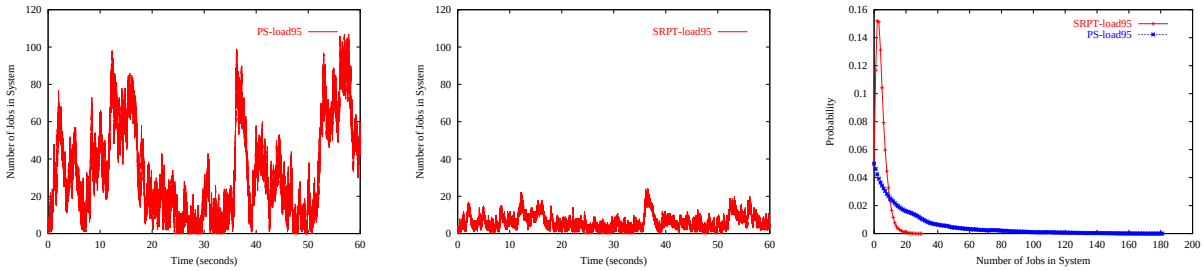
Figure 5: Illustration of Busy Period Analysis for Simulation Validation (SRPT, U = 0.95)



(a) PS (U = 0.50) (b) SRPT (U = 0.50) (c) Marginal Dist.

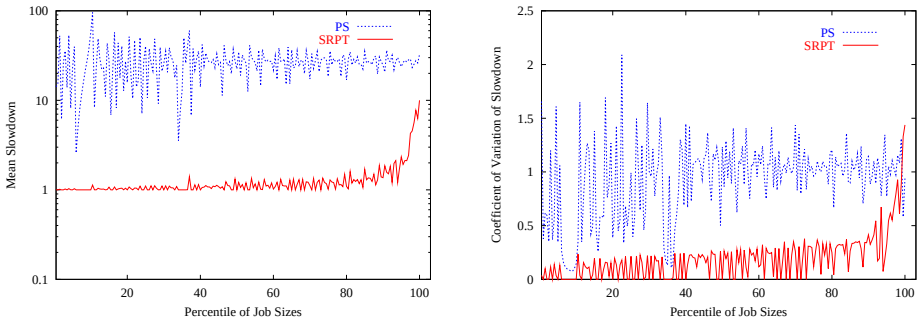


(d) PS (U = 0.80) (e) SRPT (U = 0.80) (f) Marginal Dist.



(g) PS (U = 0.95) (h) SRPT (U = 0.95) (i) Marginal Dist.

Figure 6: Simulation Results for PS and SRPT Scheduling Policies



(a) Mean Slowdown (b) CV of Slowdown

Figure 7: Illustration of Endogenous Unfairness

5.2 Revisiting Unfairness

The results in Figure 6 show that the number of jobs in the system can differ a lot from one scheduling policy to another. These results lead us to revisit the notion of "unfairness" for Web server scheduling policies. In particular, we argue that there are (at least) two different aspects to unfairness, which we call *endogenous unfairness* and *exogenous unfairness*. These are defined as follows:

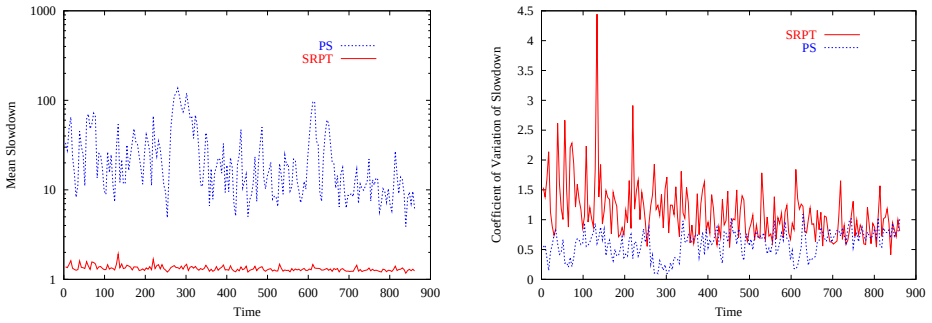
- Endogenous unfairness refers to unfairness caused by an intrinsic property of a job, such as its size. The size of a given job is the same regardless of when it arrives, and thus this aspect of unfairness is invariant.

- Exogenous unfairness refers to unfairness caused by external conditions, such as the number of other jobs in the system, their sizes, and their arrival times.

To illustrate endogenous unfairness, we study the performance of SRPT and PS on a job size basis. We grouped the jobs in the empirical trace into 200 bins according to the job sizes, and calculated the mean slowdown and the CoV of slowdown for each bin. These results are for 95% system load.

Figure 7 presents the slowdown results versus the percentile of the job size distribution. Figure 7(a) shows the mean slowdown results for PS (top line) and SRPT (bottom line). While both plots show some noise (due to exogenous unfairness effects), the PS line is roughly horizontal (as expected), while the SRPT line has a distinct upward trend in the tail of the job size distribution. That is, large jobs can experience unfairness under SRPT. The largest 1% of the jobs in SRPT experience a slowdown almost 10 times worse than that for smaller jobs, though the slowdown is still better than that with PS. Figure 7(b) shows the CoV of slowdown results for the same job size classifications. About 99% of jobs have lower CoV of slowdown with SRPT scheduling than with PS. That is, except for the largest 1% of jobs, the jobs in each bin are more fairly treated by SRPT than by PS, which suffers from exogenous unfairness.

Exogenous unfairness is due to fluctuations in ambient load. To illustrate exogenous unfairness, we study slowdown in the time domain, as a time series process. In particular, we group



(a) Mean Slowdown (b) CV of Slowdown

Figure 8: Illustration of Exogenous Unfairness

jobs into 200 bins according to the time they arrived during the simulation. In each bin, we calculate the mean slowdown, and CoV of slowdown.

Figure 8(a) shows the mean slowdown for PS and SRPT as a function of time. The mean slowdown under PS varies a lot with time, due to load variation. For example, jobs that arrive near time 300 seconds have slowdown values 10 times higher than jobs arriving near 700 seconds. In other words, bursty request arrivals can increase the slowdown dramatically for PS. The impact of load variation on SRPT in Figure 8(a) is comparatively minor.

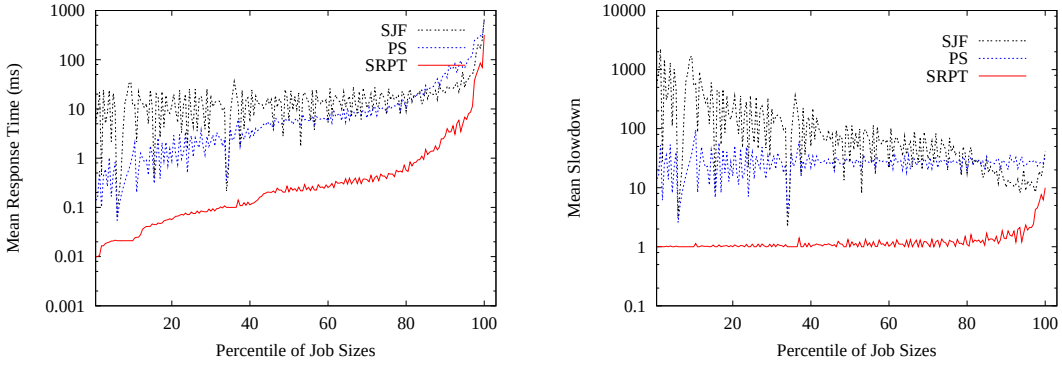
Figure 8(b) shows the results for CoV of slowdown as a function of time. Here, the CoV results for SRPT (top line) are generally higher than those for PS (lower line). That is, over any short interval of time, PS provides relatively consistent slowdown amongst jobs, while SRPT does not (because of endogenous unfairness).

The results shown in Figure 7 and Figure 8 are consistent with each other. Combined, these results support the following observations:

- The SRPT scheduling policy has high endogenous unfairness. The mean slowdown in Figure 7(a) increases with job size, and the CoV of slowdown is high in Figure 8(b).
- The SRPT scheduling policy has low exogenous unfairness. The mean slowdown in Figure 8(a) is relatively consistent, and the CoV of slowdown in Figure 7(b) is low.
- The PS policy has high exogenous unfairness. The mean slowdown varies a lot with time in Figure 8(a), and the CoV of slowdown in Figure 7(b) is high.
- The PS policy has low endogenous unfairness. The mean slowdown in Figure 7(a) is independent of job size, and the CoV of slowdown in Figure 8(b) is low.

It is important to note that pre-emption is vital to the performance of SRPT. This fact can be illustrated by considering the performance of the Shortest Job First (SJF) scheduling policy, which is the non-preemptive version of SRPT.

Figure 9 compares the results for the SJF, SRPT, and PS scheduling policies at system load 95%. Figure 9(a) shows response time results as a function of job size, while Figure 9(b) shows the



(a) Response Time (b) Slowdown

Figure 9: Simulation Results for SJF, PS, and SRPT Scheduling Policies (U=0.95)

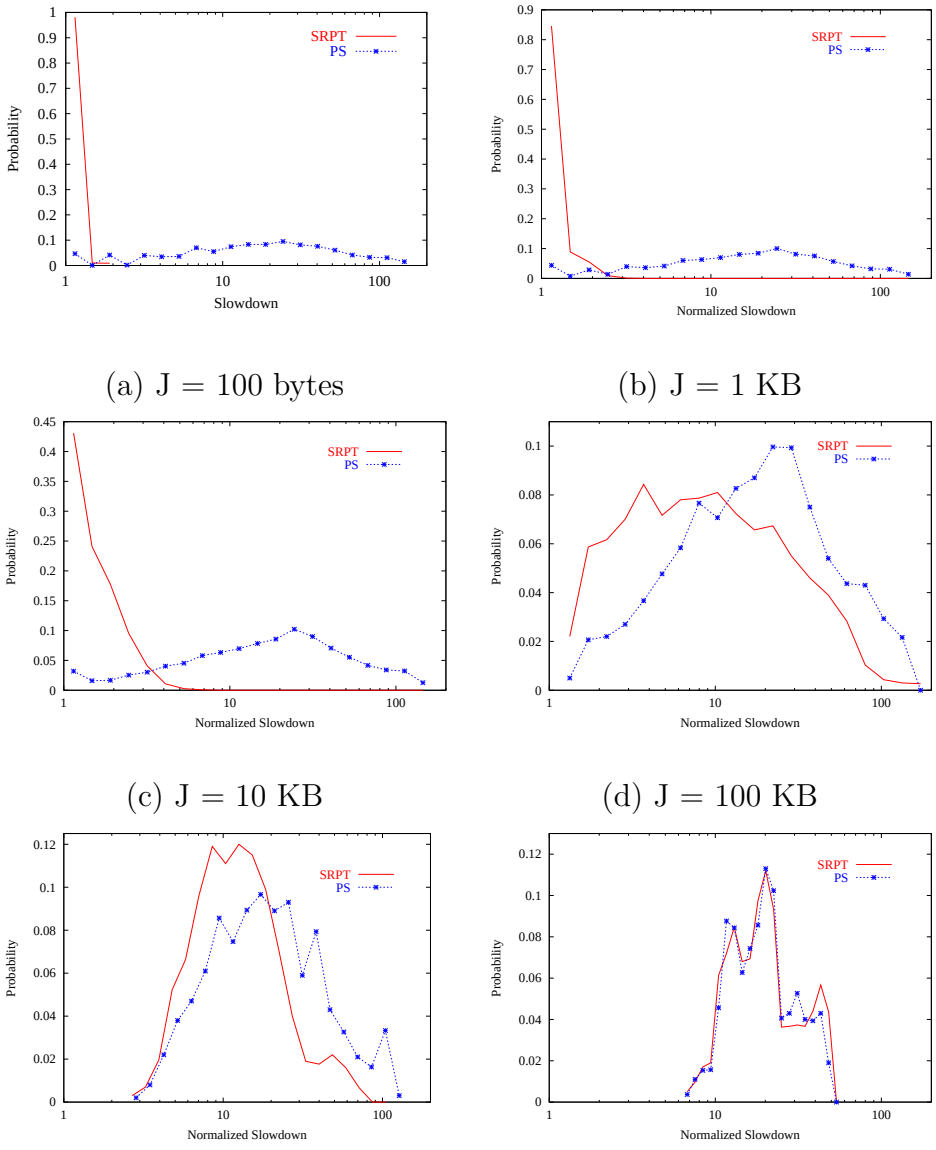
slowdown results. In general, SJF (top line) gives performance similar to the PS policy (middle line), though neither is as good as SRPT (bottom line). Interestingly, SJF performs even worse than PS for small jobs. The reason is that when system load is high, a small job can often be blocked by a large job that is already in service (a manifestation of exogenous unfairness). This greatly affects the slowdown of small jobs, even more so than it affects the response time. The obvious conclusion is that preemption is extremely important for the effectiveness of the SRPT size-based scheduling policy, because it drastically reduces exogenous unfairness.

The differences between endogenous and exogenous unfairness can be further illustrated using our simulation technique. The effect of exogenous unfairness can be quantified using our sampling methodology, by measuring the variability of response times for a particular job size, depending upon job arrival time. By varying the probe job size, system load, and scheduling policy, we can determine the expected response time behavior for a wide range of job sizes, quantifying endogenous unfairness.

Figure 10 provides a graphical illustration of the slowdown results observed for different probe job sizes. Each graph in this figure shows the marginal distribution of the slowdown metric observed from 3000 random placements of the probe job in the empirical Web server workload request stream. Note that all graphs use a logarithmic scale on the horizontal axis. These simulation results are for 95% system load.

Figure 10(a) and Figure 10(b) show the results for small probe job sizes of 100 bytes and 1 KB, respectively. For these probe job sizes, there are dramatic differences between the slowdown results for the PS and SRPT scheduling policies. For the SRPT policy, the marginal distribution is highly concentrated near 1. For the PS policy, the slowdown values span from 1 to 150.

Figure 10(c) shows the results for a 10 KB probe job size. Similar observations apply here: the SRPT policy consistently gives slowdown values below 5, while the PS policy exhibits slowdown as high as 140. Clearly, exogenous unfairness is dominant for PS, but not SRPT. Furthermore, the exogenous unfairness of PS increases with system load (see Table 3 and Figure 11).



(a) J = 100 bytes

(b) J = 1 KB

(c) J = 10 KB

(d) J = 100 KB

(e) J = 1 MB

(f) J = 10 MB

Figure 10: Sampled Marginal Distributions of Normalized Slowdown for PS and SRPT Scheduling, for Different Probe Job Sizes (U = 0.95)

Table 3: Statistical Results for Normalized Slowdown

System Load	Probe Job Size	PS Policy		SRPT Policy		Statistical Difference?	Better Policy
		Mean	Var	Mean	Var		
U = 0.50	J = 1 KB	2.06	2.42	1.05	0.03	Yes	SRPT
U = 0.50	J = 10 KB	2.07	2.13	1.23	0.11	Yes	SRPT
U = 0.50	J = 100 KB	2.06	1.17	1.85	0.88	Yes	SRPT
U = 0.50	J = 1 MB	2.04	0.31	1.94	0.16	Yes	SRPT
U = 0.50	J = 10 MB	2.01	0.04	2.00	0.04	No	-
U = 0.80	J = 1 KB	5.61	28.87	1.09	0.05	Yes	SRPT
U = 0.80	J = 10 KB	5.63	27.71	1.45	0.28	Yes	SRPT
U = 0.80	J = 100 KB	5.44	18.87	4.54	17.62	Yes	SRPT
U = 0.80	J = 1 MB	5.18	5.88	4.55	3.16	Yes	SRPT
U = 0.80	J = 10 MB	5.11	1.36	5.07	1.34	No	-
U = 0.95	J = 1 KB	27.16	844.19	1.11	0.07	Yes	SRPT
U = 0.95	J = 10 KB	27.22	828.06	1.58	0.40	Yes	SRPT
U = 0.95	J = 100 KB	27.11	801.18	15.59	391.71	Yes	SRPT
U = 0.95	J = 1 MB	26.01	582.09	15.06	130.53	Yes	SRPT
U = 0.95	J = 10 MB	21.32	94.39	21.77	115.32	No	-

As the size of the probe job is increased, the differences between the two marginal distributions are less pronounced. For example, Figure 10(d) shows the simulation results for a probe job size of 100 KB. The two marginal distributions partially overlap. According to a t-test, the means of the two distributions are statistically different (15.59 for SRPT versus 27.11 for PS), at the 0.05 level of significance (see Table 3). Under the PS policy, the *mean* slowdown is approximately the same for all probe job sizes (as expected). However, the *variance* of slowdown is a decreasing function of job size. For SRPT, the mean slowdown clearly depends on job size. Furthermore, the variance of slowdown peaks at intermediate job sizes (e.g., 100 KB); the variance is lower both for smaller jobs and for larger jobs. In other words, endogenous unfairness is dominant for SRPT; its effect is also more pronounced at higher load (see Table 3).

Figure 10(e) presents results for a probe job size of 1 MB. Here, the two distributions overlap a lot, though the SRPT policy still has a shorter tail, compared to PS. The means of the distributions still differ statistically.

Finally, Figure 10(f) presents results for a 10 MB probe job. At this job size, there is no statistical difference between the means of the two distributions. In fact, the distributions are almost visually identical. This graphical result supports the claim in [16] about the asymptotic convergence of scheduling policies with respect to slowdown (though our metric is different).

Table 4: Statistical Results for Normalized Slowdown (U = 0.95)

Probe Job Size	PS Policy		SRPT Policy		Statistical Difference?	Better Policy
	Mean	Var	Mean	Var		
J = 1 MB	26.01	582.09	15.06	130.53	Yes	SRPT
J = 2 MB	25.24	481.32	18.86	164.65	Yes	SRPT
J = 2.5 MB	24.71	422.61	26.38	671.76	Yes	PS
J = 3 MB	24.16	366.74	25.92	504.08	Yes	PS
J = 3.5 MB	23.62	312.41	25.13	428.22	Yes	PS
J = 4 MB	23.30	268.19	24.79	371.34	Yes	PS
J = 5 MB	22.38	216.65	22.86	274.84	No	-
J = 10 MB	21.32	94.39	21.77	115.32	No	-

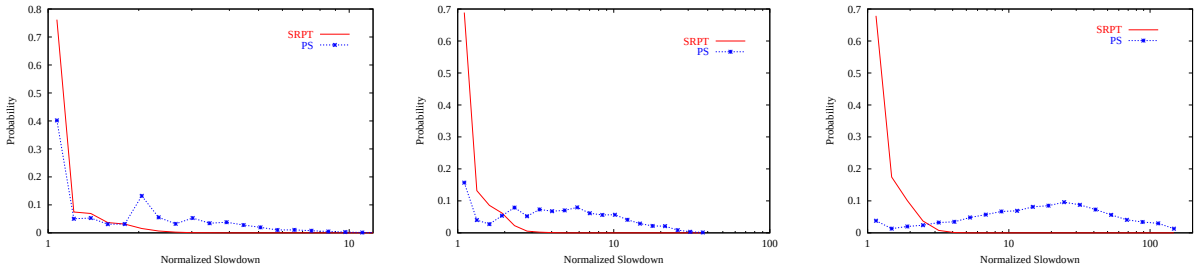
5.3 The Crossover Region

Harchol-Balter *et al.* [16] state (and prove) an intriguing theoretical claim: while the asymptotic slowdown results for the largest jobs are the same for any scheduling policy, there are (slightly smaller) large jobs for which SRPT is worse in terms of slowdown, by a factor $1 + \epsilon$, for small $\epsilon > 0$. However, their paper provides no concrete information on where this "crossover region" occurs (i.e., what job size range). As the second main contribution in this paper, we apply our sampling methodology in an attempt to find this region, for our empirical workload.

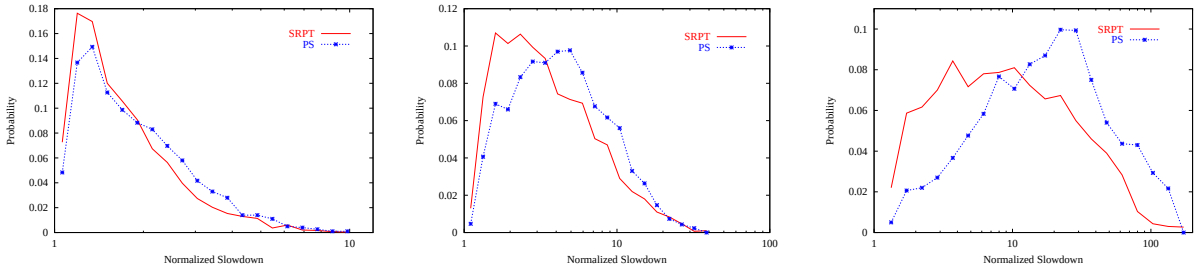
Figure 11 summarizes our simulation results. These experiments present results for probe job sizes ranging from 3 KB to 10 MB, with system loads ranging from 50% to 95%. For 50% load (first column of graphs in Figure 11) and 80% load (second column of graphs in Figure 11), no crossover effect is evident, for any of the probe job sizes considered. For 95% load (third column of graphs in Figure 11), a slight crossover effect appears in Figure 11(i). For this graph, the probe job size is 3 MB, and the system load is 95%.

To further explore this phenomenon, Figure 12 presents more detailed simulation results for 95% system load. For our particular workload trace, probe jobs in the range of 2.5-4 MB are in the "crossover region" (see Table 4). Figures 12(a) and (b) show example results for a probe job of size 4 MB. Figure 12(a) uses a linear horizontal scale, while Figure 12(b) uses a logarithmic scale. In these plots, the SRPT results show slightly longer tail behaviour than the PS results. This difference, though small, is enough to skew the mean of the distribution, leading to the crossover effect. The differences in means between SRPT and PS are statistically significant (t-test, 0.05 level of significance). Table 4 summarizes these results.

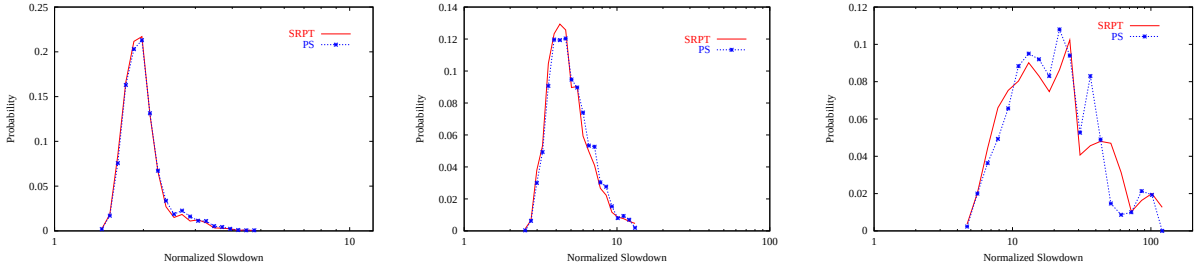
In summary, our probe-based sampling approach has provided independent verification of the "crossover region" established theoretically in [16]. Furthermore, our results have quantified its practical range for an empirical Web server workload.



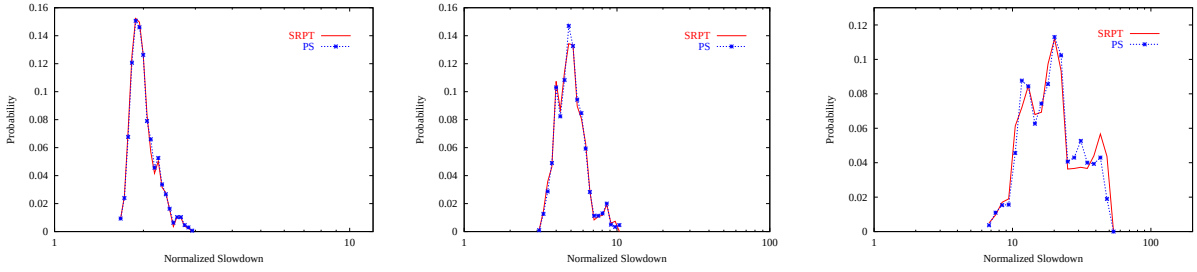
(a) J = 3 KB, U = 0.50 (b) J = 3 KB, U = 0.80 (c) J = 3 KB, U = 0.95



(d) J = 100 KB, U = 0.50 (e) J = 100 KB, U = 0.80 (f) J = 100 KB, U = 0.95

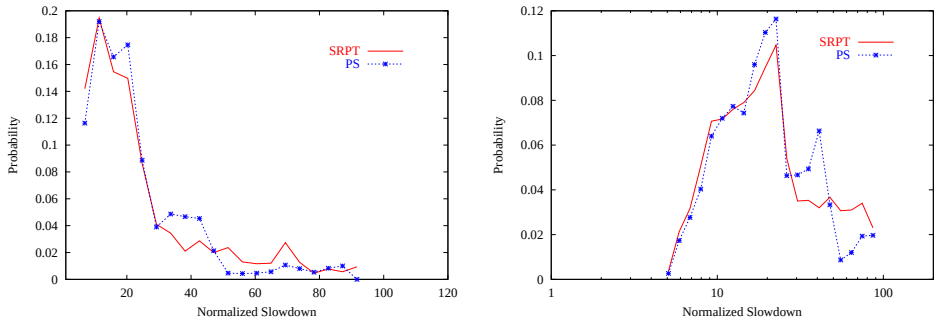


(g) J = 3 MB, U = 0.50 (h) J = 3 MB, U = 0.80 (i) J = 3 MB, U = 0.95



(j) J = 10 MB, U = 0.50 (k) J = 10 MB, U = 0.80 (l) J = 10 MB, U = 0.95

Figure 11: Simulation Results Searching for the SRPT "Crossover Region"



(a) J = 4 MB (linear scale) (b) J = 4 MB (log scale)

Figure 12: Detailed Simulation Results Illustrating the "Crossover Region" (U = 0.95)

6 Simulation Results: Synthetic Traces

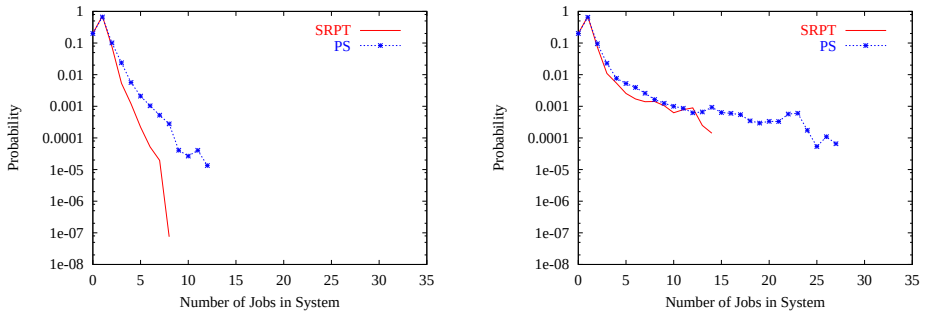
In this section, additional experiments are conducted to evaluate the sensitivity of the previous results to the request arrival process (e.g., self-similarity), and the job size distribution (e.g., heavy-tailed). Synthetic traces are generated using WebTraff [20], a locally-developed tool for modeling Web server and Web proxy workloads.

6.1 Effect of Request Arrival Process

It is well-known that network traffic exhibits the features of long-range dependence and self-similarity [13, 18, 23]. Long-range dependence (LRD) refers to non-negligible correlations that occur in the arrival count process across many time scales, from seconds to minutes to hours. Self-similarity refers to a "fractal" pattern in the traffic: similar looking bursts are seen across many time scales. Several studies have shown that self-similarity has an adverse impact on network performance due to amplified queueing delay and packet loss [23]. Self-similarity is a special type of LRD with hyperbolic decay to the autocorrelation function. Standard statistical methods exist for testing for LRD and self-similarity [23].

In order to evaluate the impact of a self-similar traffic arrival process on the performance of the scheduling policies, the effects caused by job size differences should be removed. This is achieved by using fixed size jobs (3 KB in this experiment). When the job size is constant, the slowdown for a particular job under PS scheduling depends primarily on the number of competing jobs present in the system. The SRPT policy behaves the same as FCFS: a job that arrives and begins service first will have fewer bytes remaining than any job that arrives later.

The degree of self-similarity in the synthetic traces is captured by the Hurst parameter H, which we vary from 0.50 (low) to 0.90 (high). For each trace, the self-similar arrival count process is converted into a timestamped arrival process by randomly distributing requests uniformly in each one second interval. For all traces, the average request arrival rate is the same. The performance results for the SRPT and PS policies are evaluated. The metrics used for comparison are the mean slowdown, the standard deviation of slowdown, and the CoV of slowdown.



(a) Hurst Parameter $H = 0.5$ (b) Hurst Parameter $H = 0.9$

Figure 13: Marginal Distribution of Number of Jobs in the System for Different Arrival Processes (3 KB Jobs, $U = 0.80$)

The experiments are conducted for three different load levels: 50%, 80%, and 95%. For space reasons, only the results for 80% system load are presented here.

Figure 13 shows marginal distribution plots for the number of jobs in the system for PS and SRPT scheduling. Figure 13(a) is for $H = 0.5$, while Figure 13(b) is for $H = 0.9$. Figure 13 shows that the number of jobs in the system tends to increase with the degree of self-similarity in the request arrival count process. This observation applies for both PS and SRPT. The impact of the bursty arrival process is even more pronounced at higher load.

In general, increasing the burstiness of the request arrival process (i.e., higher Hurst parameter) has an adverse impact on system performance, consistent with previous research. For both PS and SRPT policies, the mean slowdown and the variance of slowdown tend to increase with an increase in the Hurst parameter. The differences are more pronounced at higher system load. Even with a constant job size, the response time of a particular job in PS highly depends on when it arrives and the number of competing jobs during the processing. The simulation results show that PS has higher mean slowdown and higher CoV of slowdown than SRPT at different system loads. A bursty arrival process (higher values of Hurst parameter) has a larger adverse impact on the performance of PS than on SRPT.

Additional experiments show that the CoV of slowdown for the PS policy is higher than that for the SRPT policy at all system loads and for all Hurst parameters tested. Once again, this suggests that PS is more prone to exogenous unfairness than SRPT. In particular, jobs arriving in a bursty fashion are treated more unfairly by the PS policy than by the SRPT policy.

6.2 Effect of Heavy-tailed Job Size Distribution

Previous studies have investigated the distribution of file sizes seen on Web servers [2, 5, 8]. An important property of Web file size distributions is that they exhibit a heavy tail.

The heavy-tailed property indicates high variability in file size. This is often manifested as many small values (mice) mixed with a few very large values (elephants). The weight of the elephants typically skews the mean of the distribution, making it much larger than the median.

Table 5: Characteristics of Synthetic Traces for Heavy-Tailed Workloads

Item	$\alpha = 1.0$	$\alpha = 1.2$	$\alpha = 1.4$	$\alpha = 1.6$	$\alpha = 1.8$	$\alpha = 2.0$
Total Requests	96,000	96,000	96,000	96,000	96,000	96,000
Total Transferred Bytes (GB)	1.5	0.85	0.69	0.65	0.57	0.51
Smallest Transfer Size (bytes)	59	59	59	59	59	59
Median Transfer Size (bytes)	3,503	3,503	3,503	3,503	3,503	3,503
Mean Transfer Size (bytes)	15,804	8,932	7,257	6,911	5,942	5,360
Largest Transfer Size (bytes)	13,788,773	5,592,636	3,663,559	1,681,387	408,505	40,621

Heavy-tailed distributions are prevalent in the Web [2, 8].

The purpose of this experiment is to assess the impact of a heavy-tailed job size distribution on the performance of PS and SRPT scheduling policies. Similar to the idea in the previous section, here the effects caused by the burstiness of the arrival process should be removed. This is achieved by generating the timestamps for different jobs according to a deterministic arrival process (i.e., constant spacing between job arrivals). The same timestamps are used for each trace. This mitigates the effects of exogenous unfairness. Again, the simulations are conducted for 50%, 80%, and 95% system load, though we present only the results for 80% load for brevity.

The transfer size distribution is carefully controlled by adjusting the parameters in WebTraff. In WebTraff [20], the transfer size distribution in a synthetic trace is modelled using a hybrid distribution, consisting of a lognormal distribution for the body and a Pareto model for the tail. The proportion of transfers in the tail of the distribution is another parameter to the model. The default setting is 20%, with the tail beginning at $k = 10$ KB.

To investigate the impact of the heavy-tailed property only, the effect from the body of the distribution is eliminated by keeping the lognormal parameters the same for all traces. Only the file sizes larger than the 10 KB threshold are changed for traces with different Pareto parameters.

The Pareto parameter value α determines the weight of the tail. The smaller the value of α is, the more pronounced the heavy-tailed property is. The empirically observed value of α normally ranges from 1.0 to 1.6. In this experiment, a slightly larger range (1.0 to 2.0) is used.

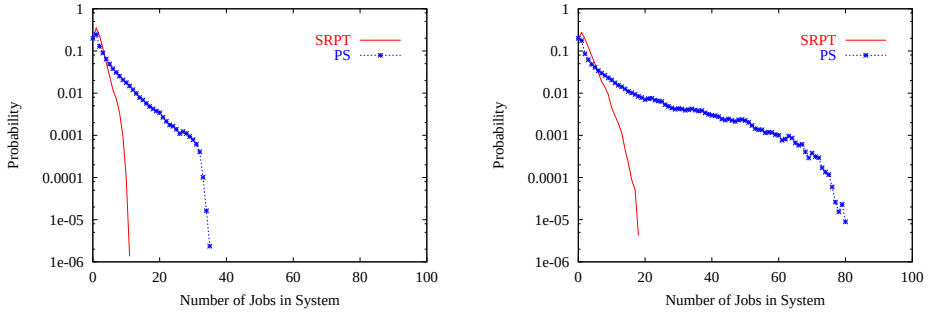
Table 5 summarizes the statistical characteristics for 6 traces used in the simulations. Table 6 summarizes the tail behaviour for the 6 traces. The results show that as α increases, the mean and standard deviation of file size decrease. Moreover, for $\alpha = 1.00$, the largest 1% of all jobs make up 58% of the total bytes transferred by the server. For comparison, for $\alpha = 2.0$, the largest 1% of all jobs account for only 6% of the total load.

Figure 14 shows the simulation results for 80% system load. Figure 14(a) shows results for the lightest heavy-tail value considered ($\alpha = 2.0$), while Figure 14(b) shows results for $\alpha = 1.4$. In both plots, the PS policy always has a larger tail to the distribution than for the SRPT policy. The length of the tail depends on the heaviness of the tail for the job size distribution (i.e., inversely related to α).

This behaviour makes sense intuitively. When the system load is low, small incoming jobs

Table 6: Statistical Analysis of Heavy-Tailed Workloads in Synthetic Traces (Tail Only)

Pareto Value	Minimum	Median	Maximum	Mean	Std Dev
$\alpha = 1.00$	9,287	18,701	13,788,773	78,400	404,111
$\alpha = 1.20$	9,728	17,023	5,592,636	37,171	125,546
$\alpha = 1.40$	9,598	15,447	3,663,559	27,119	58,514
$\alpha = 1.60$	10,037	15,199	1,681,387	25,041	40,818
$\alpha = 1.80$	10,147	13,868	408,505	19,232	22,340
$\alpha = 2.00$	10,180	13,331	40,621	15,736	6,409



(a) Pareto Parameter $\alpha = 2.0$ (b) Pareto Parameter $\alpha = 1.4$

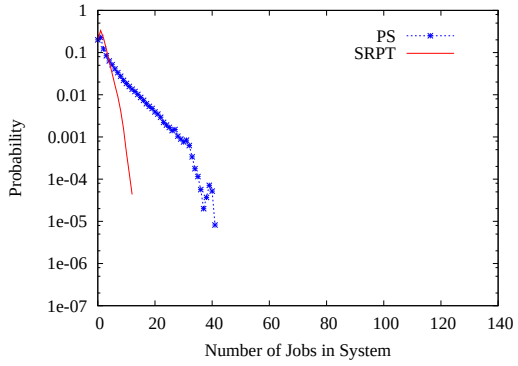
Figure 14: Marginal Distribution of Number of Jobs for Different Tail Weights ($U = 0.80$)

often find the system empty and finish soon. Backlogs occur only when a large job shows up in the system. The larger the job size is, the longer the backlog will last, resulting in more jobs in the system. This phenomenon is more pronounced for the PS policy, and less pronounced for the SRPT policy.

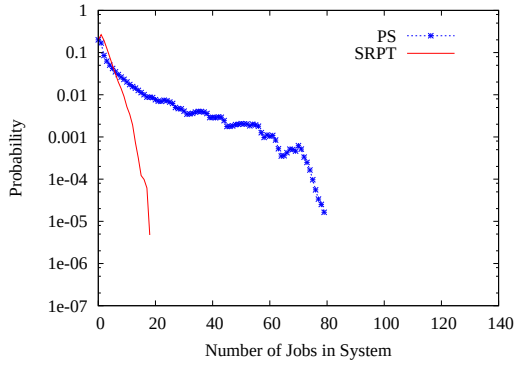
In summary, by adjusting the tail behaviour of file size distribution, the impact of heavy tails on the scheduling policies is illustrated. The low α cases represent high variability in file size distribution, whereas the high α cases represent low(er) variability in the file size distribution. The CoV of slowdown for SRPT tends to be an increasing function of α. SRPT has a greater advantage over PS when the heavy-tailed property is present. In fact, endogenous unfairness is more pronounced when variability in the file size distribution is low.

6.3 Combined Effects

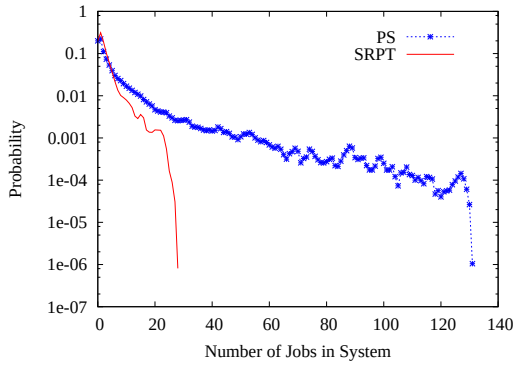
As a final simulation experiment, we consider the combined effects of LRD traffic and heavy-tailed job size distributions. The simulations are conducted using four new synthetic traces. The four traces are generated by combining two different arrival processes and two different job size distributions. For arrival processes, we consider SRD ($H = 0.5$) and LRD ($H = 0.9$) processes. For job size distributions, we consider heavy-tailed distributions with two different tail weights.



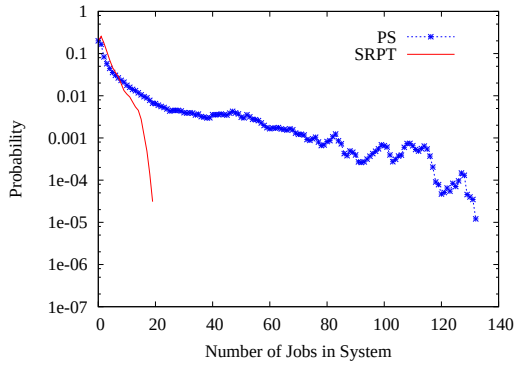
(a) SRD ($H = 0.5$), "Lighter" Tail ($\alpha = 2.0$)



(b) SRD ($H = 0.5$), "Heavier" Tail ($\alpha = 1.4$)



(c) LRD ($H = 0.9$), "Lighter" Tail ($\alpha = 2.0$)



(d) LRD ($H = 0.9$), "Heavier" Tail ($\alpha = 1.4$)

Figure 15: Marginal Distributions for Combined Effects of LRD and Heavy-tails (U=0.80)

Both are generated using a Pareto distribution for the tail, with different tail index α. We use $\alpha = 2.0$ for a "lighter" heavy tail, and $\alpha = 1.4$ for a "heavier" heavy tail. We mix the two different arrival processes with the two different heavy-tailed distributions to get four different traces.

Figure 15 shows the results from these experiments. Each graph shows the marginal distribution for the number of simultaneously active jobs in the system, under SRPT and PS policies. Figure 15(a) shows the "best case" for the four workloads considered ($H = 0.5$, $\alpha = 2.0$), while Figure 15(d) shows the "worst case" ($H = 0.9$, $\alpha = 1.4$),

The results in Figure 15 are quite consistent with those in the previous two subsections. That is, the more bursty the arrival process, the worse the impact on the performance of the scheduling policies. Similarly, the heavier the tail of the job size distribution, the worse the impact on the performance of the scheduling policies.

The main new observation here is that the arrival process has a larger impact on the performance of the scheduling policies than does the tail characteristics of the job size distribution. This is illustrated by comparing Figures 15(b) and (c). Also, we can observe the impact from different job size distributions for $H = 0.5$ (as in Figures 15(a) and (b)). However, the difference is negligible when there is a bursty arrival pattern (as in Figures 15(c) and (d)). In other words, the arrival process has a more dominant effect than the job size distribution, when comparing scheduling policies.

While most analytical work has focused on M/M/1 or M/G/1 models, our results show that it is very important to study the scheduling policy behaviour under a bursty arrival process.

7 Summary and Conclusions

This paper has presented a detailed study of the unfairness properties of the PS and SRPT Web server scheduling policies. Our work is carried out using trace-driven simulation, with an empirical workload trace from World Cup 1998.

The paper makes three main contributions:

- *The paper revisits and refines the notion of unfairness.*

  Two types of unfairness in a Web server scheduling system are identified: *endogenous unfairness* that a job can suffer because of its own size, and *exogenous unfairness* that a job can suffer because of the state of the Web server (i.e., other jobs in the system) at the time it arrives. The simulation results show that the SRPT scheduling policy has higher endogenous unfairness than the PS policy, while the PS policy has higher exogenous unfairness.

- *The paper confirms prior theoretical results regarding the crossover region and asymptotic convergence, illustrating these properties for an empirical workload.*

  A probe-based sampling methodology was developed and used for estimating the mean and variance of slowdown for different Web server scheduling policies. The approach is general-purpose, in that it can be applied for any arrival process, service time distribution,

and scheduling policy. By applying this approach, the asymptotic convergence property of the slowdown metric is illustrated for the largest jobs, providing independent confirmation of previous theoretical results [16]. Furthermore, the existence of the "crossover region" is verified for some job sizes under SRPT scheduling, again confirming prior theoretical results [16], and extending them to an empirical workload.

- *The paper studies the impact of the request arrival process and the job size distribution on the performance of PS and SRPT.*

  A sensitivity study is conducted using synthetic traces with desired workload characteristics. The simulation results illustrate three properties. First, as the degree of self-similarity (burstiness) in the request arrival process increases, system performance worsens for both PS and SRPT. However, the effect on PS is more pronounced, because of exogenous unfairness. In general, a bursty request arrival process tends to reduce the size of the crossover region. Second, the heavier the tail of the job size distribution is, the greater the performance differences between PS and SRPT. The SRPT policy thrives on the heavy-tailed property, which ameliorates its endogenous unfairness. Finally, when the combined effects of traffic arrival process and heavy-tailed distributions are considered, our results show that the bursty traffic arrival process has a more pronounced effect.

We believe that our simulation-based approach complements the theoretical and experimental work in the literature on SRPT. We hope that our results provide further insight into unfairness, increasing the "comfort level" associated with SRPT scheduling, and encouraging its deployment in Internet Web servers.

References

[1] M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site", *IEEE Network*, Vol. 14, No. 3, pp. 30-37, May/June 2000.

[2] M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 631-645, 1997.

[3] B. Avi-Itzhak and H. Levy, "On Measuring Fairness in Queues", *Advances in Applied Probability*, Vol. 36, No. 3, pp. 919-936, 2004.

[4] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness", *Proceedings of ACM SIGMETRICS Conference*, Cambridge, MA, pp. 279-290, June 2001.

[5] P. Barford and M. Crovella, "Measuring Web Performance in the Wide Area", *ACM Performance Evaluation Review*, August 1999.

[6] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection Scheduling in Web Servers", *Proceedings of USENIX Symposium on Internet Technologies and Systems* (USITS), Boulder, CO, October 1999.

[7] M. Crovella, M. Harchol-Balter, and S. Park, "The Case for SRPT Scheduling in Web Servers", Technical Report MIT-LCS-TR-767, MIT Laboratory for Computer Science, October 1998.

[8] M. Crovella, M. Taqqu, and A. Bestrvros, "Heavy-Tailed Probability Distributions in the World Wide Web", *A Practical Guide To Heavy Tails*, Ch. 1, pp. 3-26, Chapman and Hall, New York, 1998.

[9] E. Friedman and S. Henderson, "Fairness and Efficiency in Web Server Protocols", *Proceedings of ACM SIGMETRICS Conference*, San Diego, CA, pp. 229-237, June 2003.

[10] M. Gong, *Quantifying Unfairness in Web Server Scheduling*, M.Sc. Thesis, Department of Computer Science, University of Calgary, July 2003.

[11] M. Gong and C. Williamson, "Quantifying the Properties of SRPT Scheduling", *Proceedings of IEEE/ACM MASCOTS*, Orlando, FL, pp. 126-135, October 2003.

[12] M. Gong and C. Williamson, "Simulation Evaluation of Hybrid SRPT Scheduling Policies", *Proceedings of IEEE/ACM MASCOTS*, Volendam, Netherlands, pp. 355-363, October 2004.

[13] M. Grossglauser and J-C. Bolot, "On the Relevance of Long-Range Dependence in Network Traffic", *Proceeedings of ACM SIGCOMM'96*, Stanford, CA, pp. 15-24, August 1996.

[14] E. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks", *IEEE Journal on Selected Areas in Communication*, Vol. 9, No. 7, pp. 1024-1039, 1991.

[15] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based Scheduling to Improve Web Performance", *ACM Transactions on Computer Systems*, Vol. 21, No. 2, pp. 207-233, May 2003.

[16] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic Convergence of Scheduling Policies with Respect to Slowdown", *Proceedings of IFIP Performance 2002*, Rome, Italy, pp. 241-256, September 2002.

[17] Internet Traffic Archive, http://ita.ee.lbl.gov/

[18] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking*, 2(1): 1-15, 1994

[19] D. Lu, H. Shen, and P. Dinda, "Size-based Scheduling Policies with Inexact Scheduling Information", *Proceedings of IEEE/ACM MASCOTS*, Volendam, Netherlands, pp. 31-38, October 2004.

[20] N. Markatchev and C. Williamson, "WebTraff: A GUI for Web Proxy Cache Workload Modeling and Analysis", *Proceedings of IEEE MASCOTS*, Fort Worth, TX, pp. 356-363, October 2002.

[21] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. Gehrke, "Online Scheduling to Minimize Average Stretch", *IEEE Symposium on Foundations of Computer Science*, pp. 433-442, 1999.

[22] E. Nahum, M. Rosu, S. Seshan, and J. Almeida, "The Effects of Wide-Area Conditions on WWW Server Performance", *Proceedings of ACM SIGMETRICS Conference*, Cambridge, MA, pp. 257-267, June 2001.

[23] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling". *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pp. 226-244, 1994.

[24] R. Perera, "The Variance of Delay Time in Queueing System M/G/1 with Optimal Strategy SRPT", *Archiv für Elektronik und Übertragungstechnik*, Vol. 47, pp. 110-114, 1993.

[25] I. Rai, G. Urvoy-Keller, and E. Biersack, "Analysis of LAS Scheduling for Job Size Distributions with High Variance", *Proceedings of ACM SIGMETRICS Conference*, San Diego, CA, pp. 218-238, June 2003.

[26] D. Raz, H. Levy, and B. Avi-Itzhak, "RAQFM: A Resource Allocation Queueing Fairness Measure", *Proceedings of ACM SIGMETRICS Conference*, New York, NY, June 2004.

[27] R. Schassberger, "The Steady-State Appearance of the M/G/1 Queue under the Discipline of Shortest Remaining Processing Time", *Advanced Applied Probability*, Vol. 22, pp. 456-479, 1990.

[28] L. Schrage, "A Proof of the Optimaility of the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 16, pp. 678-690, 1968.

[29] L. Schrage and L. Miller, "The Queue M/G/1 with the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 14, pp. 670-684, 1966.

[30] F. Schreiber, "Properties and Applications of the Optimal Queueing Strategy SRPT: A Survey", *Archiv für Elektronik und Übertragungstechnik*, Vol. 47, pp. 372-378, 1993.

[31] B. Schroeder and M. Harchol-Balter, "Web Servers Under Overload: How Scheduling Can Help", to appear, *ACM Transactions on Internet Technology*, 2005.

[32] D. Smith, "A New Proof of the Optimaility of the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 26, pp. 197-199, 1976.

[33] A. Wierman and M. Harchol-Balter, "Classifying Scheduling Policies with Respect to Unfairness in an M/GI/1", *Proceedings of ACM SIGMETRICS Conference*, San Diego, CA, pp. 238-249, June 2003.