Quantifying the Properties of SRPT Scheduling

Mingwei Gong   Carey Williamson
Department of Computer Science
University of Calgary

Abstract

This paper uses a probe-based sampling approach to study the behavioural properties of Web server scheduling strategies, such as Processor Sharing (PS) and Shortest Remaining Processing Time (SRPT). The approach is general purpose, in that it can be used to estimate the mean and variance of the job response time, for arbitrary arrival processes, scheduling policies, and service time distributions.

In the paper, we apply the approach to trace-driven simulation of Web server scheduling to compare and contrast the PS and SRPT scheduling policies. We identify two types of unfairness, called *endogenous* and *exogenous* unfairness. We quantify each, focusing on the mean and variance of slowdown, conditioned on job size, for a range of system loads. Finally, we confirm recent theoretical results regarding the asymptotic convergence of scheduling policies with respect to slowdown, and illustrate typical performance results for a practical range of job sizes from an empirical Web server workload.

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 [2, 3, 5]. 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 that remaining for 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 [12, 15]. Under overload, SRPT also minimizes the number of jobs starved [2].

The primary concern with SRPT is that a large job in the system may be delayed indefinitely if the continuous arrival of smaller jobs preempts it from service. Prior results in the literature clearly establish the response time 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 [2, 5].

Harchol-Balter *et al.* [6] have recently established asymptotic bounds on the slowdown (defined as the job response time divided by the job size) for the largest jobs under SRPT (or any other) scheduling policy. In particular, their results show that slowdown asymptotically converges to the *same* value for *any* pre-emptive work-conserving scheduling policy (even Longest Remaining Processing Time, LRPT). 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 under SRPT is worse than under PS by at most a factor of $1 + \epsilon$, for small $\epsilon > 0$. In our paper, we use the term "crossover effect" to refer to the region where SRPT provides worse performance than PS.

The foregoing theoretical results for SRPT motivate our present paper, which focuses on the performance "in practice" for typical job sizes at an SRPT Web server. There are two specific issues that we address. First, the theoretical results mentioned previously hold only for "sufficiently large" jobs. It is not clear what "sufficiently large" means in practice. (Of course, this may be workload dependent.) Second, is the crossover effect observable in practice? If so, for what range of job sizes does it occur?

In this paper, we use a probe-based sampling approach to evaluate job slowdown for SRPT and PS scheduling policies in a simulated Web server system. We use trace-driven simulation with empirical Web request streams from the 1998 World Cup Web site in an attempt to quantify the behavioural properties of SRPT scheduling.

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. This approach enables a methodical study of the performance differences between SRPT and other scheduling policies. The approach is general-purpose, in that it can be used for arbitrary arrival processes and service time distributions. The approach is not limited to the M/G/1 queue assumptions, for instance. Furthermore, the approach can be used to estimate the mean and variance of the response time for any scheduling policy, even those for which no closed-form analytical solution is known.

One of the main observations from our experiments is 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) at the time it arrives. By quantifying these effects separately, we provide new insights into the differences between SRPT and PS.

Our simulation results show significant performance advantages for SRPT for small (e.g., 1-10 KB) and medium size (e.g., 100 KB to 1 MB) jobs. The slowdown results for the SRPT and PS policies asymptotically converge for the largest job sizes considered (e.g., 10 MB). These results are consistent with the theoretical work in [6]. Finally, we apply the sampling methodology in an attempt to observe the "crossover effect" in practice. The results for our Web server workload show that at high load (95-99%), job sizes in the range of 2.5-4 MB experience slightly worse performance under SRPT scheduling than under PS.

The remainder of this paper is organized as follows. Section 2 discusses background information on Web server performance and SRPT scheduling. Section 3 explains the probe-based sampling methodology. Section 4 describes the experimental methodology for our simulation study. Section 5 presents the simulation results and analyses. Section 6 summarizes 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, 6, 9]. 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 shares resources equally amongst contending requests.

The Shortest Remaining Processing Time (SRPT) policy optimizes mean job response time. Using job size information, the SRPT policy selects for service the job that has the least remaining service time. With this approach, the system throughput (i.e., job completions per second) is maximized, and mean job response time is minimized.

2.2 Related Work

The classic theoretical work on SRPT scheduling in queueing systems was done over 30 years ago [12, 13], and has seen renewed activity in the last 10 years [10, 11, 14]. SRPT is provably optimal in terms of mean job response time [12, 15].

The investigation of SRPT scheduling for Web servers began about 5 years ago [3]. There is theoretical work [2, 6], as well as a prototype implementation of SRPT scheduling in the Apache Web server [5]. Experimental results confirm many of the performance advantages of SRPT scheduling established in the theoretical work.

Despite this work in the research literature, concerns remain about the unfairness of SRPT scheduling. The SRPT policy is not yet widely deployed in Internet Web servers, in part because there is incomplete understanding of its behaviour for a realistic Web workload. It is on this front that our paper makes its main contribution.

3 Sampling Methodology

This section explains the probe-based sampling methodology used for studying Web server scheduling policies. Section 3.1 presents a simple example to provide some insight into the dynamic behaviour of scheduling policies. Section 3.2 explains the sampling method itself.

3.1 Preliminaries: Understanding SRPT Scheduling

Figure 1 illustrates the basic issues related to Web server request scheduling. On the left in Figure 1 is a Web server workload. This simple example has 20 requests. The two-column format shows the timestamp (in seconds) and the job (response) size in bytes for each request. This trace format is assumed for Web server workloads throughout the paper.

The graphs in Figure 1 illustrate the dynamic busy period structure for four different Web server scheduling policies (FCFS, PS, SRPT, and LRPT) at a Web server processing this workload. We assume that the Web server is idle when the first request arrives.

Figure 1(a) shows the instantaneous number of jobs in the system for the FCFS policy on this workload, as a function of time. The unit size vertical upward steps represent job arrivals, and the vertical downward steps represent job departures. Figure 1(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 1(c) shows the instantaneous number of jobs in the system for the PS policy on the same workload, and Figure 1(d) shows the corresponding number of bytes. Figures 1(e) and (f) show the corresponding results for the SRPT policy, while Figures 1(g) and (h) show the results for LRPT.

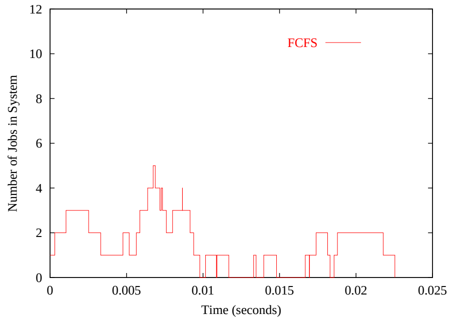
Three observations are evident from Figure 1. First, while the times at which job departures occur are different, the plots for "byte backlog" are identical for each of the scheduling policies considered. This (obvious) property holds for *any* work-conserving scheduling policy, assuming the same job 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 the first observation, and is again obvious. What this means is that the number of busy periods, as well as the mean and variance of the busy period duration is *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 number of jobs simultaneously in the system is *different* for each of the policies considered. For example, on the sample workload illustrated in Figure 1, the SRPT policy never has more than 3 jobs simultaneously in the system, while the FCFS and PS policies each have 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 policies 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[1] of the competing jobs.

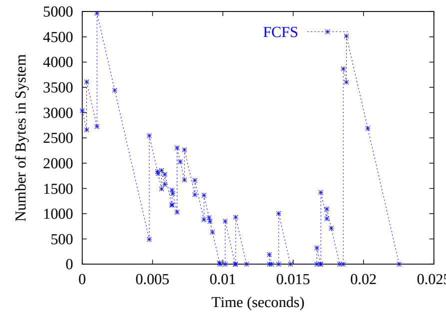
The difference in "jobs in the system" is our focus in this paper. The probe-based sampling methodology (described next) estimates the impact of this property on job response time.

[1]Intuition suggests that the fewer competing jobs in the system, the sooner service will be received, but this is not necessarily true for SRPT: it depends on job size. Furthermore, if the pending jobs tend to be large, an arriving job may be serviced soon, regardless of the number of jobs in the system.

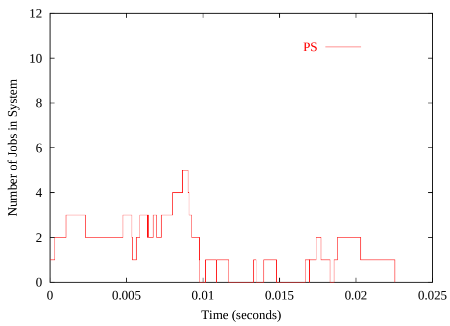
Time	Bytes
0.000000	3038
0.000315	949
0.001048	2240
0.004766	2051
0.005642	366
0.005872	201
0.006380	298
0.006742	1272
0.007271	597
0.008008	283
0.008653	482
0.010165	852
0.010911	929
0.013306	191
0.013969	1005
0.016681	322
0.016961	1420
0.017391	191
0.018563	3867
0.018783	914



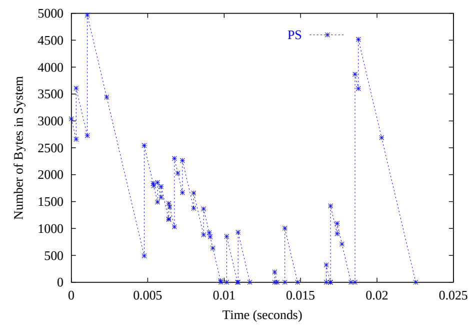
(a) Jobs in System (FCFS)



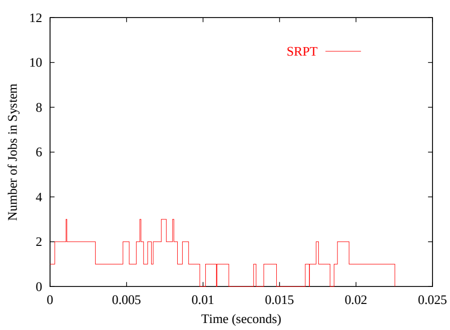
(b) Backlog in Bytes (FCFS)



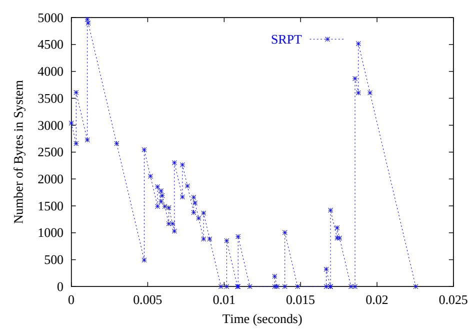
(c) Jobs in System (PS)



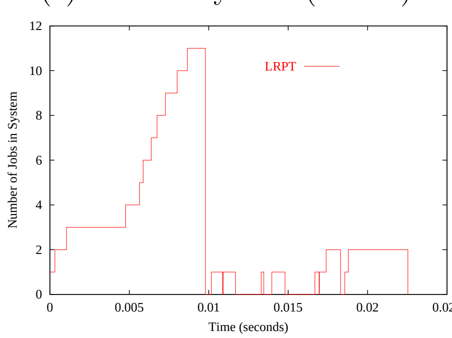
(d) Backlog in Bytes (PS)



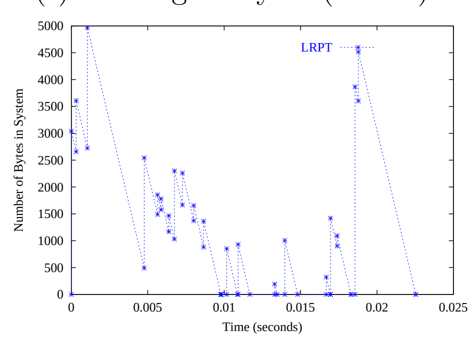
(e) Jobs in System (SRPT)



(f) Backlog in Bytes (SRPT)



(g) Jobs in System (LRPT)



(h) Backlog in Bytes (LRPT)

Figure 1: Simulation Results Illustrating Busy Period Structure for Four Scheduling Policies

```
For scheduling algorithm S = ( FCFS, PS, SRPT, LRPT, ... ) do
   For background load level U = ( 0.50, 0.80, 0.95 ) do
     For probe job size J = ( 1 B, 10 B, 100 B, 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 slowdown metric 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 2: Algorithmic Overview of Sampling Methodology Using Probe Jobs

3.2 Probe-based Sampling Algorithm

Figure 2 provides a high-level description of the sampling methodology for quantifying the properties of SRPT 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 Web workload stream and a scheduling policy at the Web server, a single *probe job* is inserted uniformly at random into the request arrival stream. The Web server is then 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 same probe job, we obtain an estimate of the response time distribution for a job of that size. By repeating the experiment with different probe job sizes, we assess the characteristics (e.g., mean response time, variance, fairness, unfairness) of a specific scheduling policy. Varying the system load (e.g., by setting the network link capacity) determines when unfairness is most pronounced, for a particular job size.

A naive implementation of the algorithm in Figure 2 would be compute-intensive, requiring many executions of the Web server simulator, each with a slightly modified request stream. There are several ways 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.

Our current implementation of the algorithm uses a checkpointing technique so that all job probes are 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 Experimental Methodology

4.1 Simulation Model and Assumptions

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 simulation assumes that the Web server deals only with static Web content, for which response size is known by the server. A fluid-flow approximation is assumed for network transfers, so that service time is proportional to job size. Outgoing network bandwidth is assumed to be the bottleneck. We ignore propagation delays, packetization issues, and other network effects. We also assume that the context switch cost[2] for the server is zero.

The Web server model in the simulation is simple. A configuration parameter specifies the service rate for the server, in bytes per second. A second configuration parameter specifies the scheduling policy to be used. Currently, our simulator supports FCFS, PS, SRPT and LRPT. 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 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 Web server workload used in our experiments[3] is an empirical trace from the 1998 World Cup Web site [1, 7]. The trace has 1 million requests, representing just over 14 minutes of server activity. The arrival process is stationary over this interval, with an average rate of 1160 requests per second. The largest 1% of the transfers account for 20% of the bytes transferred.

Table 1 provides further information about the trace. In our experiments, we augment the one-second resolution timestamps in the original trace to distribute requests randomly (uniformly) across each one-second interval, while preserving the order of arrivals.

[2]This assumption favours the PS scheduling policy, which can potentially have an unbounded number of context switches, depending on the server quantum size chosen (e.g., 1 byte). The maximum number of context switches is bounded for policies such as FCFS and SRPT (N and $2N$, respectively, for N job arrivals [2]).

[3]We have also conducted similar experiments with synthetic traces [4]. The results are qualitatively similar, and too numerous to report here.

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

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, as indicated in Figure 2.

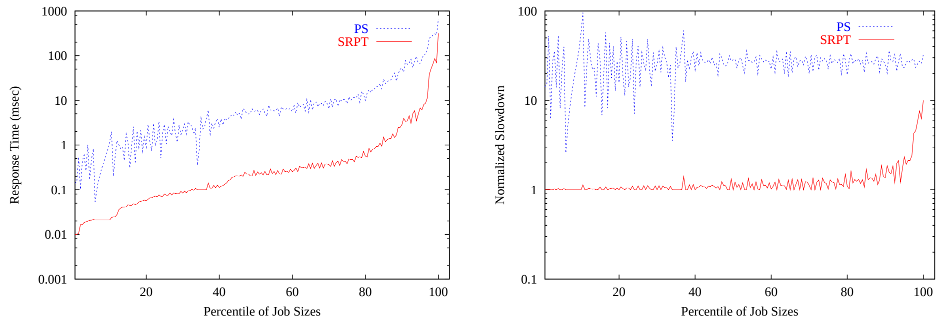
The scheduling policies considered[4] are PS and SRPT. System load is controlled by setting the byte service rate (i.e., network link capacity) for the server. We consider probe job sizes ranging from 1 byte to 10 MB, since this spans the typical range of Web object sizes. The largest probe job size considered (10 MB) represents a "small" perturbation to the total load (i.e., it increases the total transferred bytes by less than 0.3%). However, it is larger than any other job size in the empirical workload, and depending on the probe placement, could extend the simulated completion time slightly.

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 in time series plots and in frequency histogram plots to illustrate the behaviours of different scheduling policies. It is also 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.

[4]For space reasons, this paper reports only a subset of the experiments. Full results appear in [4].



(a) Response Time (b) Slowdown

Figure 3: Simulation Validation Results for PS and SRPT Scheduling Policies (U = 0.95)

- *Slowdown*: 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 [6, 8]. The slowdown value ranges from 1 to infinity. Lower values of slowdown represent better 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.* The CoV of slowdown indicates the variability (or unfairness) of slowdown across jobs. In a perfectly fair environment, the CoV of slowdown should be zero. The larger the CoV value, the higher the degree of unfairness.

4.5 Validation

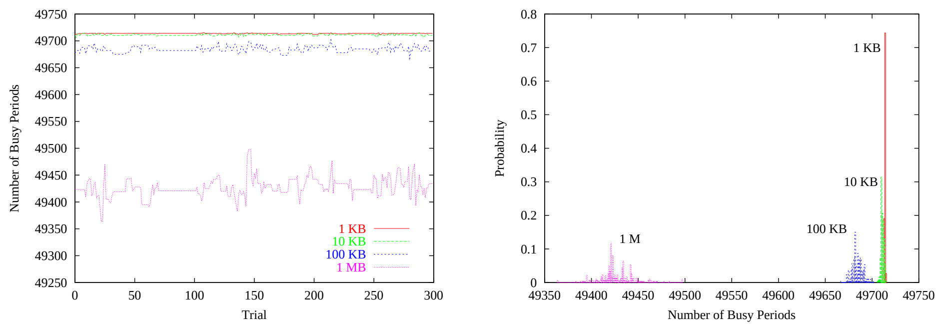
Significant effort was expended on verification and validation of the results reported by our simulator. This section briefly describes several of these steps.

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 comparing 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 3, for our workload trace. Figure 3(a) shows job response time, while Figure 3(b) shows the slowdown metric, both plotted versus the percentile of the job size distribution, following the format used in [2]. Our simulation results are qualitatively consistent[5] with those reported in [2], providing further confidence in the results reported by our simulator.

The third validation step involved testing our sampling approach to ensure that it followed the PASTA principle. The Anderson-Darling goodness of fit test was used to test for exponentiality,

[5]The extra "jitter" in our plots is attributed to the transfer size distribution in the World Cup trace.



(a) Busy Periods (b) Marginal Distribution

Figure 4: Illustration of Busy Period Analysis for Simulation Validation (SRPT, U = 0.95)

while autocorrelation tests were used to test for independence. Our probe generation approach passed both tests, indicating that the system state is sampled in a Poisson fashion.

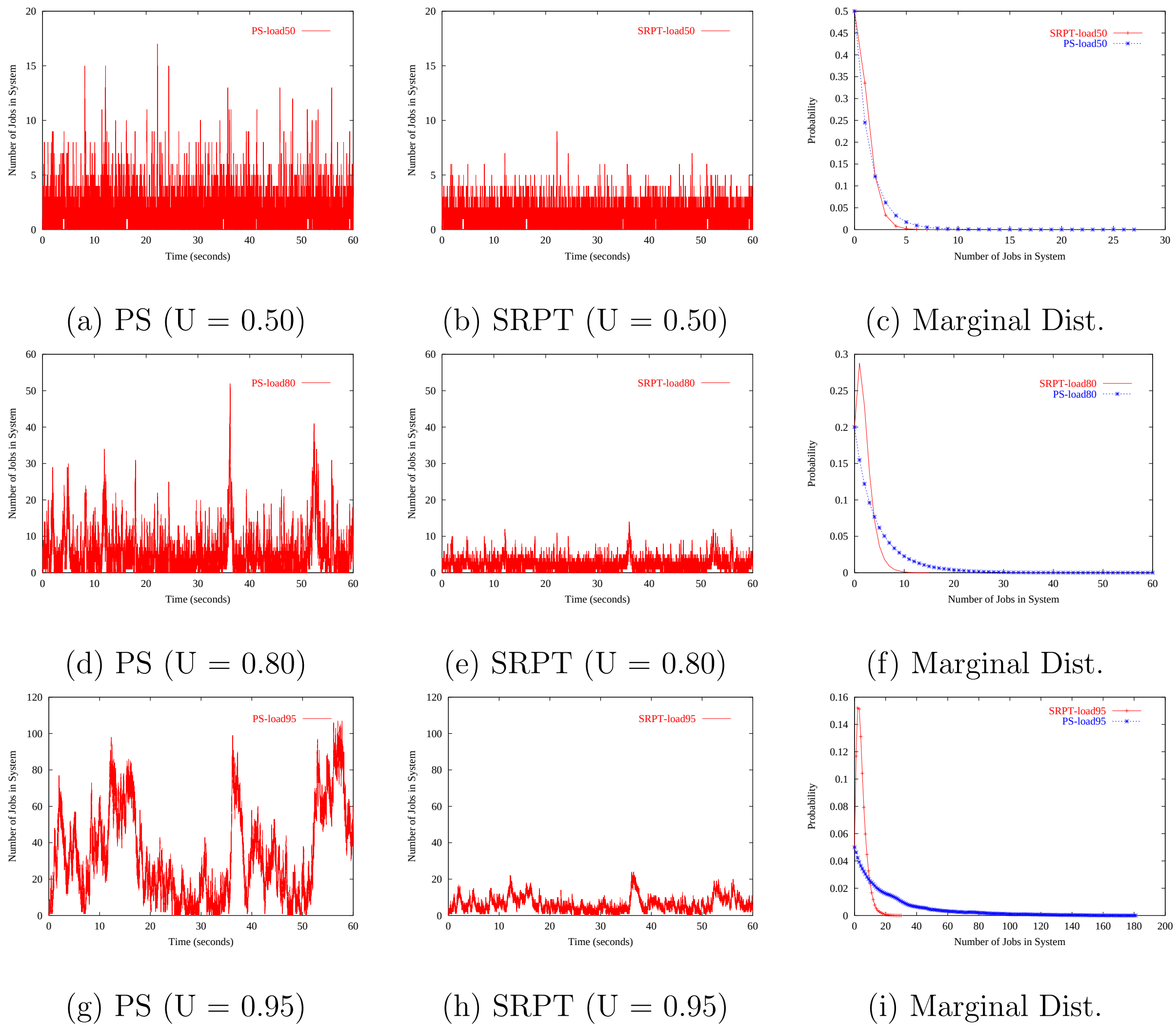
One additional validation test studied the number of busy periods, and how this number is transformed by the insertion of the probe job. Three cases are possible:

- *The probe job increases 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 leaves 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 reduces 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 4 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, and removes at most two busy periods (compared to the initial total of 49,714 busy periods). 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.



(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 5: Simulation Results for PS and SRPT Scheduling Policies

5 Simulation Results

5.1 General Observations

Figure 5 illustrates the key differences between the PS and SRPT policies. The first two columns of graphs show short (60 second) time series plots for the number of jobs simultaneously in the system for PS and SRPT, respectively, on the empirical Web server workload trace. The third column shows the marginal distributions (frequency histograms) of the number of jobs in the system, based on the entire World Cup trace. The results are illustrated for three different levels of system load: 50%, 80%, and 95%.

The top row of graphs in Figure 5 shows the results for 50% load. The time series plots show the dynamic number of jobs in the system for each scheduling policy: PS in Figure 5(a), and SRPT in Figure 5(b). Figure 5(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. The visual evidence suggests that the PS policy has a longer tail to the distribution, but it is not clear if this is statistically significant. At this modest level of load, it is rare to have more than 10 jobs in the system, with either policy.

The second row of graphs in Figure 5 shows the results for 80% load. Again, two time series plots are shown (PS in Figure 5(d), and SRPT in Figure 5(e)), with the marginal distribution results in Figure 5(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 5, for 95% load. In Figure 5(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.

5.2 Refining the Notion of Unfairness

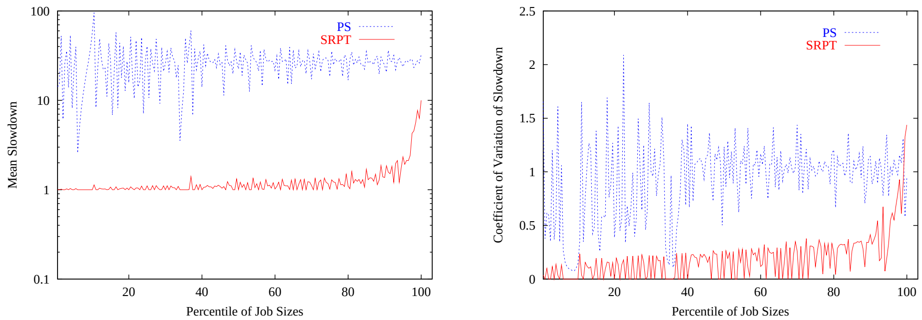
The results in Figure 5 show that the number of jobs in the system can differ a lot from one scheduling policy to another. One implication of these results relates to "unfairness" in Web server scheduling policies. In particular, we argue that there are two types of unfairness, which we call *endogenous 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.
- 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. These external factors are beyond the control of an arriving job.

To illustrate endogenous unfairness, we present in Figure 6 the mean and CoV of slowdown as a function of job size. Jobs are classified by size into 200 bins, each representing one-half of one percentile of the job size distribution [5]. These results are for 95% system load.

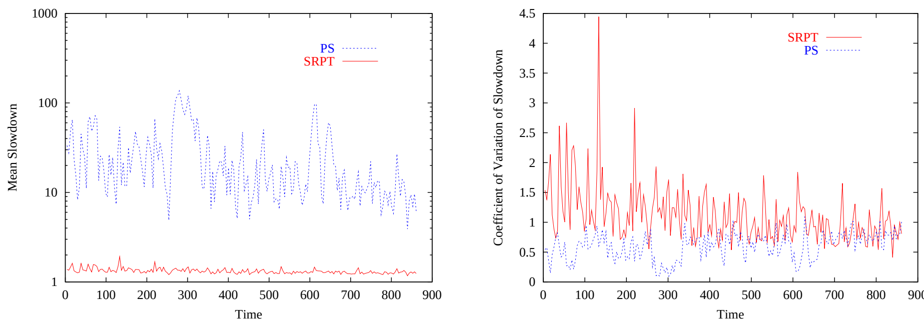
Figure 6(a) shows the mean slowdown results for SRPT and PS as a function of job size. For SRPT, the largest 1-2% of job sizes experience slowdown almost 10 times worse than smaller jobs. However, they still receive faster service under SRPT than under PS. Figure 6(b) shows the CoV of slowdown for these two policies, again as a function of job size. Within each job size classification, SRPT provides consistent slowdown, with very low CoV. About 99% of jobs have lower CoV of slowdown under SRPT than under PS. Except for the largest 1% of files, the requests in each size-based bin are more equally treated under SRPT than under PS.

Exogenous unfairness, on the other hand, arises from the dynamic arrivals and departures of jobs in the system as a whole. Requests that arrive at busy times experience relatively longer waiting times under PS than under SRPT. To illustrate exogenous unfairness, we plot in Figure 7



(a) Mean Slowdown (b) CoV of Slowdown

Figure 6: Illustration of Endogenous (Size-based) Unfairness



(a) Mean Slowdown (b) CoV of Slowdown

Figure 7: Illustration of Exogenous (Time-based) Unfairness

the mean and CoV of slowdown for SRPT and PS, with requests classified into 200 bins based on similar arrival times. For each bin, we calculate the mean and CoV of slowdown.

Figure 7(a) shows the mean slowdown as a function of time. For PS, the mean slowdown varies a lot, due to the random load variation. For example, jobs arriving near 300 seconds experience slowdown 10 times worse than jobs arriving near 700 seconds. On the other hand, the influence of load variation on SRPT is negligible.

Figure 7(b) shows the CoV of slowdown versus time. Here, the CoV of slowdown for SRPT is higher than for PS most of the time. In other words, in any short time interval, pending requests are more unfairly treated under SRPT than under PS.

The results in Figure 6 and Figure 7 are complementary. Together, they show that:

- *SRPT has high endogenous unfairness, but low exogenous unfairness.* The mean slowdown increases with job size (Figure 6(a)), but does not vary much with time (Figure 7(a)). The CoV of slowdown is consistently low with respect to job size in Figure 6(b), but varies a lot with respect to job arrival time (Figure 7(b)).

- *PS has high exogenous unfairness, but low endogenous unfairness.* The mean slowdown varies a lot with time in Figure 7(a), though it is quite consistent across job sizes (Figure 6(a)). The CoV of slowdown is high with respect to size in Figure 6(b), but relatively consistent over short intervals of time (Figure 7(b)).

5.3 Quantifying Unfairness

Exogenous unfairness can be quantified using our sampling methodology, since it allows us to measure 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 determine the expected response time for a wide range of job sizes, quantifying endogenous unfairness.

Figure 8 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. Table 2 summarizes the results for three loads.

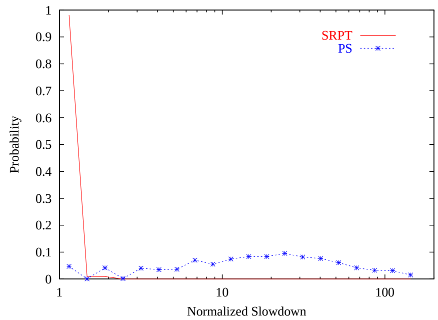
Figure 8(a) shows the results for a small probe job of size 100 bytes. For this probe job size, there is a dramatic difference between the slowdown results for the PS and SRPT scheduling policies. For the SRPT policy, the marginal distribution is highly concentrated near 1.0, the optimal value. For the PS policy, the slowdown values span a wide range, up to 150.

Figure 8(b) shows the results for a 1 KB probe job size. Similar observations apply here: the SRPT policy consistently gives slowdown values below 3, while the PS policy exhibits slowdowns as large as 150. Clearly, exogenous unfairness is dominant for PS, but not SRPT. Furthermore, the exogenous unfairness of PS increases with system load (see Table 2).

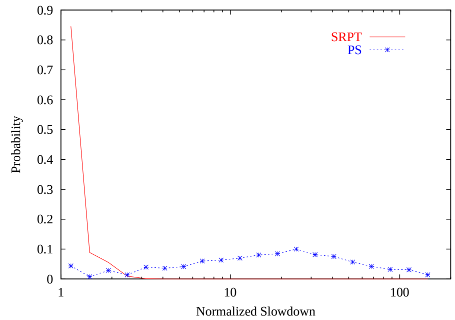
As the size of the probe job is increased, the differences between the two marginal distributions are less pronounced. For example, Figure 8(d) shows the simulation results for a probe job size of 100 KB. Here, the two marginal distributions partially overlap. According to a t-test, the means of the two distributions are statistically different (15.59 for SRPT versus 27.11 for PS), at the 0.05 level of significance (see Table 2). 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 with SRPT 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 2).

Figure 8(e) presents results for a probe job size of 1 MB. Here, the two distributions overlap significantly, though the SRPT policy still has a shorter tail, compared to PS. The means of the distributions still differ statistically.

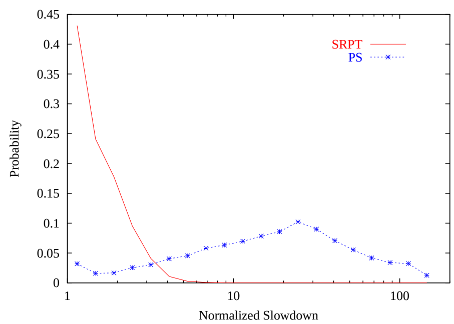
Finally, Figure 8(f) presents results for a 10 MB probe job. At this job size, there is no statistical difference between the means of the two distributions. In fact, the distributions are almost visually identical. This graphical result supports the claim in [6] about the asymptotic convergence of scheduling policies with respect to slowdown.



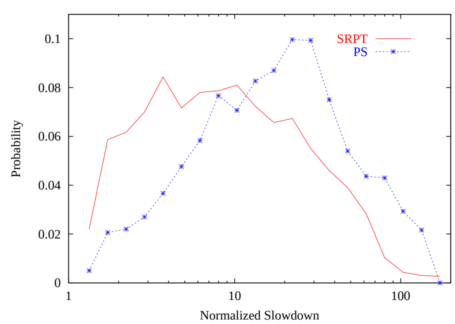
(a) J = 100 B



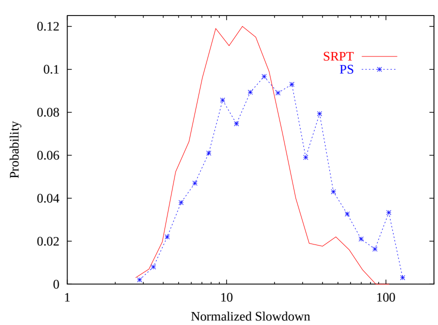
(b) J = 1 KB



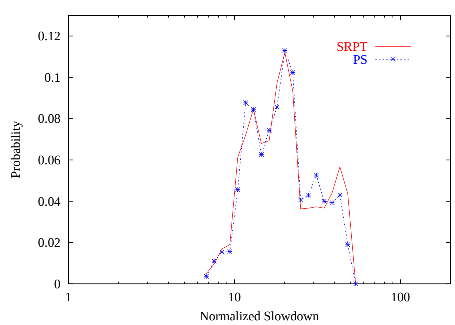
(c) J = 10 KB



(d) J = 100 KB



(e) J = 1 MB



(f) J = 10 MB

Figure 8: Sampled Marginal Distributions of Slowdown for PS and SRPT Scheduling, for Different Probe Job Sizes (U = 0.95)

Table 2: Statistical Results for 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	-

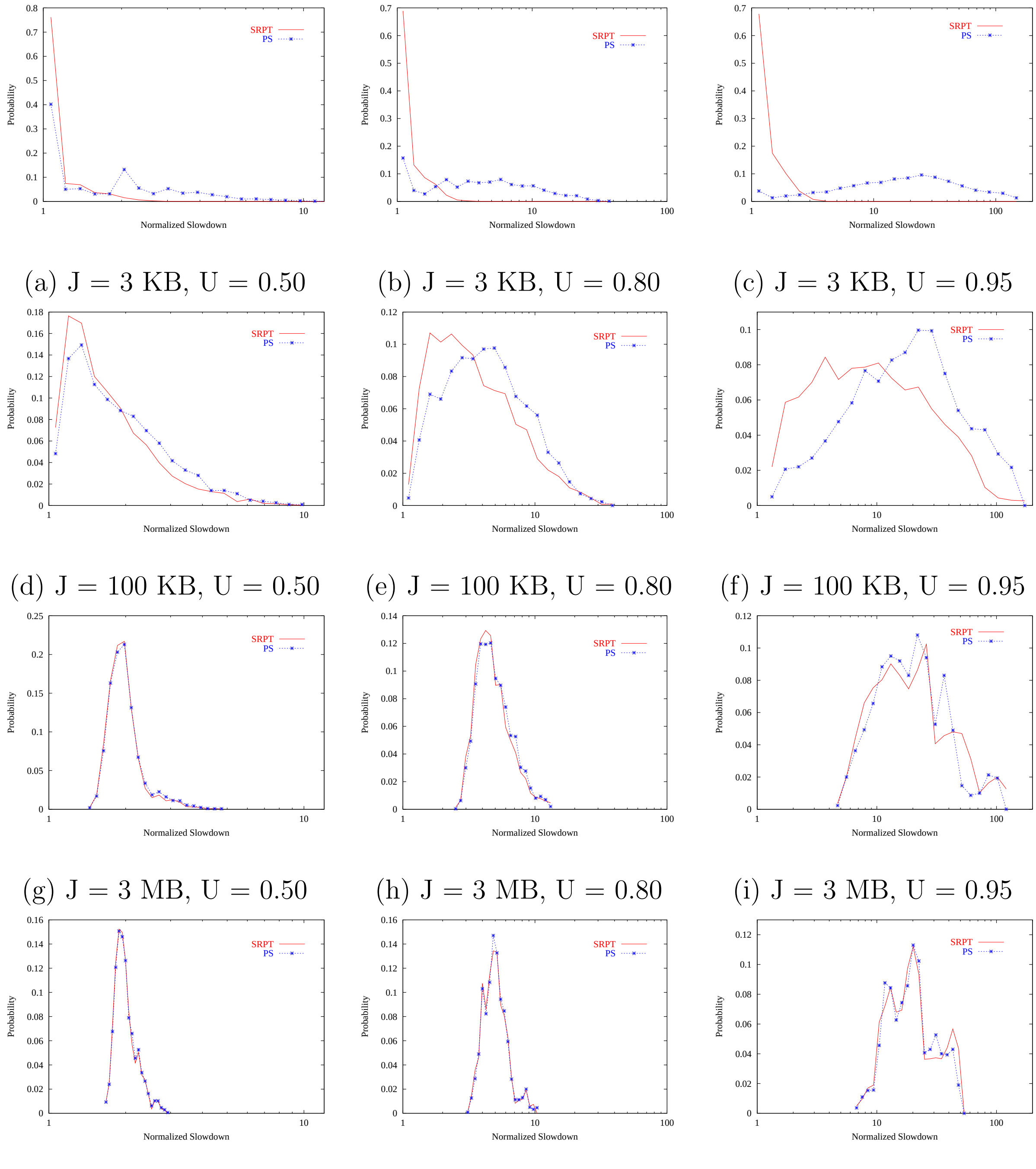
5.4 The Crossover Effect

Harchol-Balter *et al.* [6] state (and prove) an intriguing theoretical claim: while the asymptotic slowdown results for the largest jobs are the same for any scheduling policy, there is a class of (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 exactly where this "crossover" effect occurs (i.e., what job size range). As a final step in this paper, we apply our sampling methodology in an attempt to find this region, for our empirical workload.

Figure 9 shows 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 9) and 80% load (second column of graphs in Figure 9), no crossover effect is evident, for any of the probe job sizes considered. For 95% load (third column of graphs in Figure 9), a slight crossover effect appears in Figure 9(i). For this graph, the probe job size is 3 MB, and the system load is 95%.

To further explore this phenomenon, Figure 10 and Table 3 present more detailed simulation results. For our particular workload trace, at system load above 95%, we have found that probe jobs in the range[6] of 2.5-4 MB are in the "crossover" region. For example, Figure 10(a) and (b) show the results for a probe job of size 3 MB. For clarity of presentation, Figure 10(a) uses a linear horizontal scale, while Figure 10(b) uses a logarithmic scale. Figure 10(c) and (d) show

[6] The location of the crossover is also consistent with Figure 6(b), which shows the largest 2% of jobs in SRPT have higher CoV of slowdown than under PS.



(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 9: Simulation Results Searching for the SRPT "Crossover Effect"

Table 3: Statistical Results for 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	-

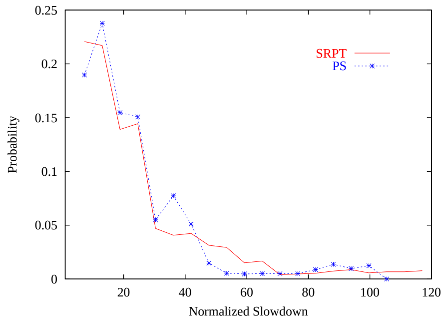
the results for a probe job of size 3.5 MB, while Figure 10(e) and (f) show the results for a 4 MB probe job. In all three pairs of plots, the SRPT results show slightly longer tail behaviour than the PS results. This difference, though small, is enough to skew the mean of the distribution, leading to the crossover effect. The differences in means between SRPT and PS are statistically significant (t-test, 0.05 level of significance). Table 3 summarizes these results.

In summary, our probe-based sampling approach has provided independent verification of the "crossover effect" established theoretically in [6]. Furthermore, our results have quantified its practical range for an empirical Web server workload.

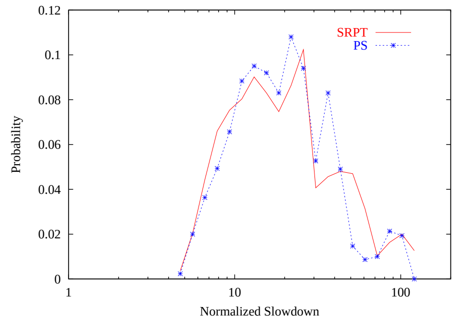
6 Summary and Conclusions

This paper describes a probe-based sampling methodology for estimating the mean and variance of job response time for Web server scheduling strategies. The approach is general-purpose, in that it can be applied for any arrival process, service time distribution, and scheduling policy. We used the approach to illustrate the asymptotic convergence of slowdown for the largest jobs, providing independent confirmation of previous theoretical results [6]. We also illustrated the existence of the "crossover effect" for some job sizes under SRPT scheduling, again confirming prior theoretical results [6]. Finally, we have quantified aspects of SRPT performance "in practice" for typical job sizes, refining the notion of unfairness (endogenous versus exogenous), and identifying the range of job sizes for which the crossover effect is evident on an empirical Web server workload.

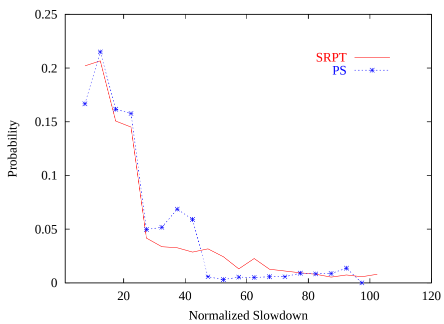
We believe that our simulation-based approach is complementary to the theoretical and experimental work in the literature on SRPT. We hope that our results provide further insight into unfairness, increasing the "comfort level" associated with SRPT scheduling, and encouraging its deployment in Internet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



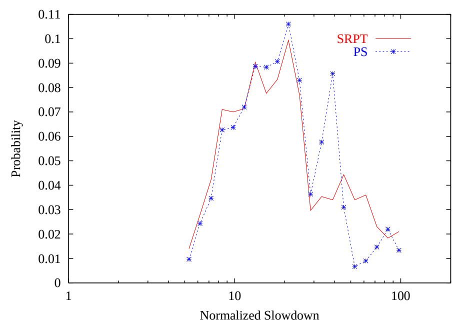
(a) J = 3 MB (linear scale)



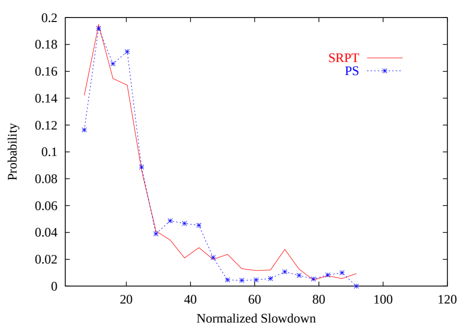
(b) J = 3 MB (log scale)



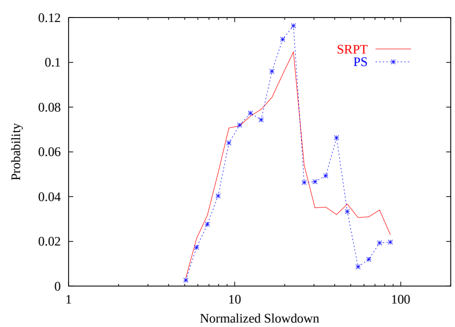
(c) J = 3.5 MB (linear scale)



(d) J = 3.5 MB (log scale)



(e) J = 4 MB (linear scale)



(f) J = 4 MB (log scale)

Figure 10: Detailed Simulation Results Illustrating the "Crossover Effect" for 3-4 MB Probe Jobs (U = 0.95)

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

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

[4] M. Gong, "Exploring Unfairness in SRPT Scheduling", M.Sc. Thesis, Department of Computer Science, University of Calgary, May 2003.

[5] M. Harchol-Balter and N. Bansal, "Implementation of SRPT Scheduling in Web Servers", Technical Report CMU-CS-00-170, CMU, October 2000.

[6] 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.

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

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

[9] 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.

[10] 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.

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

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

[13] 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.

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

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