Simulation Evaluation of Hybrid SRPT Scheduling Policies

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

Abstract

This paper uses trace-driven simulations to evaluate two novel Web server scheduling policies called K-SRPT and T-SRPT. K-SRPT is a multi-threaded version of SRPT (Shortest Remaining Processing Time) that can have up to K jobs in service at a time. T-SRPT is a hybrid policy that dynamically switches from PS (Processor Sharing) to SRPT when the number of jobs in the system exceeds T, and back to PS when the load diminishes. Slowdown profile plots are used to analyze the performance of these policies relative to PS, SRPT, and FSP (Fair Sojourn Protocol). Simulation results show that these new parameterized policies offer smooth performance tradeoffs between SRPT and PS. Stability issues for T-SRPT are also discussed.

Keywords: Web Server Performance, Scheduling, Trace-Driven Simulation, Performance Analysis

1 Introduction

Response time is important to Web users, and queueing delay at the Web server is one component contributing to the user-perceived response time. Depending on the Web workload characteristics and the server capacity, queueing delay can constitute a significant part of the overall delay.

Scheduling policies at the Web server determine the relative order for servicing client requests. Most Web servers use Processor Sharing (PS), or an approximation thereof, to provide fair service to multiple clients. It is well known that the SRPT policy minimizes mean response time [11]. However, the SRPT policy is rarely used in practice, typically for fear of unfairness and job starvation. Intuitively, it is easy to understand why SRPT seems unfair. When many small jobs arrive in the system, a large job may receive no service. Unfairness concerns plague SRPT-like policies, which give precedence to short jobs.

There is inherent tension between fairness and response time in Web server scheduling. Improving performance for one metric often comes at the expense of performance for the other metric. This type of tradeoff is well-documented in the literature [3, 5, 6]. Fairness of scheduling policies is an important current topic in networking research [10, 12].

In this paper, we propose two novel Web server scheduling policies called K-SRPT and T-SRPT. K-SRPT is a multi-threaded version of SRPT allowing up to K jobs in service at a time. T-SRPT is a hybrid policy that dynamically switches between PS and SRPT depending on the number of jobs in the system. A single parameter governs each policy's operation.

Trace-driven simulations are used to evaluate these policies, comparing their performance to PS, SRPT, and FSP (Fair Sojourn Protocol) [5]. The simulation experiments use a probe-based sampling methodology developed in previous work [6]. The simulation results show that the new scheduling policies provide a smooth tradeoff between the responsiveness of SRPT and the fairness of PS. Practical issues regarding the parameterization of T-SRPT are also discussed.

The rest of the paper is organized as follows. Section 2 discusses related work on Web server scheduling. Section 3 describes the methodology used in our work. Section 4 presents baseline results for PS, SRPT, and FSP. Section 5 presents results for the K-SRPT and T-SRPT policies. Section 6 concludes the paper.

2 Related Work

Previous results for scheduling policies show that biasing towards small jobs improves mean response time and mean slowdown. The SRPT policy has been well-studied both analytically [3, 8] and practically [4, 7].

Friedman and Henderson [5] propose a new scheduling policy called the Fair Sojourn Protocol (FSP). FSP computes the times at which jobs would complete under PS and then orders the jobs in terms of earliest PS

completion times. FSP then devotes full service to the uncompleted job with the earliest PS completion time. FSP provides performance that is provably no worse than PS for any sample path of jobs [5].

The fairness issue has recently received more and more attention in the literature. Harchol-Balter *et al.* [8] proved asymptotic bounds on slowdown for the largest jobs under SRPT. They show that slowdown asymptotically converges to the *same* value for *any* pre-emptive work-conserving scheduling policy. In addition, they prove that for *sufficiently large* jobs, the slowdown performance under SRPT is only marginally worse than under PS. We use the term "crossover effect" to refer to this intermediate region where SRPT provides worse performance than PS [6].

Two recent studies further explore fairness for scheduling strategies [10, 12]. Wierman *et al.* [12] classify scheduling policies with respect to unfairness, identifying examples of policies within each of three classes. Raz *et al.* [10] propose a new fairness metric called Resource Allocation Queueing Fairness Measure (RAQFM) that balances job seniority and service time requirements.

3 Research Methodology

Our work tackles Web server scheduling using trace-driven simulation. This section provides further information on our research methodology.

3.1 Web Server Workload Trace

Table 1 provides a statistical summary of the empirical and synthetic Web server workloads used in our simulations. Each trace has 1 million requests, with an average transfer size of 3.5 KB.

Table 1. Summary of Trace Characteristics

Item Description	Empirical Trace	Synthetic Trace
Trace Duration	860.9 sec	860.2 sec
Total Requests	1,000,000	1,000,000
Unique Web Objects	5,549	N/A
Total Transferred Bytes	3.5 GB	3.5 GB
Smallest Transfer Size	4	6
Median Transfer Size	889	2,427
Largest Transfer Size	2,891,887	49,191
Mean Transfer Size	3,498	3,501
Standard Deviation	18,815	3,502

The primary workload used is an empirical trace from the 1998 World Cup Web site [1]. It represents the Web server activity at the host Web site for a high-profile international sports event. The selected trace has 1 million requests, representing an elapsed time duration of just over 14 minutes. This is the same trace used in our previous work [6].

Figure 1 provides more information about the empirical trace. Figure 1(a) shows a time series plot of the number of requests observed per one second interval. The average request arrival rate is 1160 requests per second. The plot shows that the arrival process is stationary during the trace. That is, the mean and variance of the arrival process do not change significantly over the 14-minute period observed.

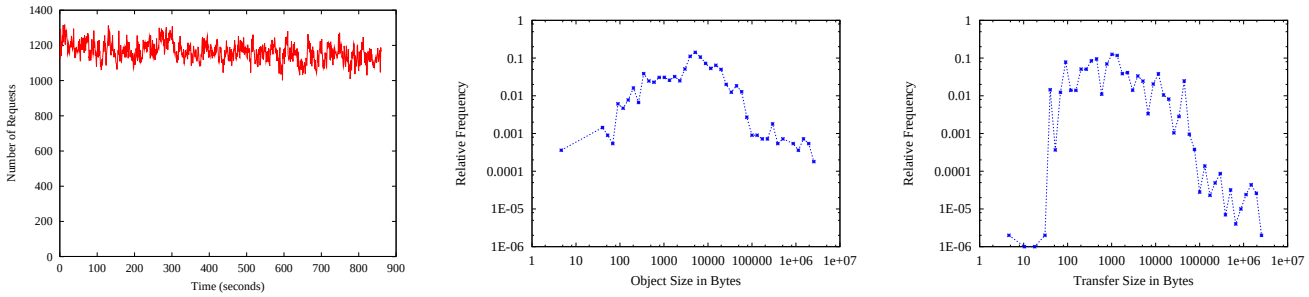
Figure 1(b) shows the size distribution for the 5,549 distinct Web objects in the workload, while Figure 1(c) shows the size distribution for the 1,000,000 transfers. The two distributions differ, since some objects are transferred more than others [2]. Figure 1(c) suggests some evidence of a heavy-tailed workload property. The largest 1% of the transfers account for 20% of the bytes transferred.

A second workload trace is used in selected experiments for sensitivity analysis. This trace is synthetically generated, assuming a Poisson arrival process and an exponential job size distribution. The synthetic trace has the same average request arrival rate as the empirical trace, and the same mean transfer size, but much lower variability in the job size distribution.

3.2 Probe-based Sampling Algorithm

In previous work, we developed a probe-based sampling methodology to evaluate Web server scheduling policies using trace-driven simulation [6]. We use the same methodology here. Given a scheduling policy at the Web server, a probe job of known size is inserted at a random point in the request arrival stream. The Web server is simulated using the modified request stream to determine the response time for the probe job. By repeating the simulation N times ($N = 3000$ in our work), we estimate the response time distribution for that probe job size.

This sampling methodology provides a robust means for comparing scheduling policies with respect to response time, slowdown, and fairness. The approach is general-purpose, suitable for arbitrary arrival processes and service time distributions. It 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.



(a) Request Profile (b) Object Size Distribution (c) Transfer Size Distribution

Figure 1. Selected Workload Characteristics for Empirical Trace (World Cup 1998)

3.3 Experimental Design

The simulation experiments use a multi-factor experimental design. The primary factors are the scheduling policy and the job size. The system load is fixed at 95% for all of the experiments. Table 2 summarizes the factors and levels used in the trace-driven simulation experiments.

We consider two main performance metrics:

- *Slowdown*: The slowdown metric is defined 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 [8, 9]. The slowdown metric quantifies the performance for different probe job sizes in the sampling methodology. We present these results in a *slowdown profile plot* that shows the mean slowdown performance versus probe job size.
- *Number of jobs in the system*: This metric illustrates the system behaviours for different scheduling policies. We use marginal distribution (frequency histogram) plots to show the distribution of number of jobs in the system for each policy.

4 Baseline Comparisons

This section lays the foundation for our work by providing a baseline comparison between the PS, SRPT, and FSP policies. We use slowdown profile plots to compare and contrast these policies with respect to mean slowdown for different probe job sizes. We use synthetic and empirical traces to understand the sensitivity of slowdown results to the request arrival process and the job size distribution.

Figure 2 presents the results from our baseline simulation experiments. Four different workloads are used here. Figure 2(a) at the top presents results for the synthetic workload with Poisson arrivals and exponentially distributed job sizes. This is the least realistic of the four workloads. Figure 2(d) at the bottom presents results for the empirical trace, with a bursty request arrival process and heavy-tailed job size distribution. This is the most realistic of the four workloads, since it represents an empirical trace. The other two workloads represent auxiliary traces produced by using a Poisson arrival process with the empirical job sizes (Figure 2(b)), and the empirical arrival process with exponentially distributed job sizes (Figure 2(c)). These four workloads allow us to understand the impacts of the arrival process and the job size distribution on scheduling policy performance.

Figure 2 illustrates several properties of each scheduling policy considered. These are described as follows:

- **PS**: The theoretical results for PS indicate an expected slowdown of $\frac{1}{1-\rho}$ for all jobs, independent of job size, for an M/M/1 system at load ρ. This result corresponds to a horizontal line in a slowdown profile plot. For the synthetic trace in Figure 2(a), the match is good: a horizontal line appears at slowdown 20 for 95% load. This result is expected, since the workload is consistent with the M/M/1 assumptions. For the other workloads, this property does not hold. In particular, the burstiness of the empirical arrival process in Figure 2(c) and Figure 2(d) leads to higher mean slowdown values, near 30 for smaller job sizes. Bursty arrivals cause a transient increase in the number of jobs in the system, reducing the per-job processing rate, and inflating the slowdown. This represents *exogenous unfairness* [6]. The PS policy is highly sensitive to the request arrival process, and somewhat less sensitive to the job size distribution.

Table 2. Experimental Factors and Levels for Simulation Study of Web Server Scheduling

Factor	Levels
System Load (ρ)	0.95
Probe Job Size (J)	1 KB, 5 KB, 10 KB, 50 KB, 100 KB, 500 KB, 1 MB, 3 MB, 5 MB
Scheduling Policy (S)	PS, FSP, SRPT, K-SRPT, T-SRPT, DT-SRPT

- **SRPT**: The SRPT policy is provably optimal for overall mean response time. Figure 2(a) illustrates three known properties of SRPT: (1) much lower slowdown values than PS for small job sizes; (2) a "crossover" region with worse slowdown than PS for some job sizes; and (3) asymptotic convergence to the same slowdown value for the largest job sizes. While these properties are clearly evident in Figure 2(a) and Figure 2(c), they are less evident (but still present) in Figure 2(b) and Figure 2(d) for the empirical job size distribution. The main observation is that SRPT is sensitive to the job size distribution. In particular, SRPT performs well on workloads with heavy-tailed job sizes. The crossover region, while still present, is small for the empirical workload. Only a small percentage of jobs experience worse slowdown under SRPT than under PS, and the average slowdown for these jobs is only marginally worse than under PS. The crossover region is located near the upper 1-2% of the job size distribution (e.g., from 2.5 MB to 4 MB in the empirical trace [6], and starting at 50 KB in the synthetic trace). Simply stated, SRPT performs better than PS for most (if not all) job sizes in realistic workloads [8].

- **FSP**: The FSP policy is provably never worse than the PS policy on any sample path [5]. This property is clearly illustrated in our slowdown profile plots: the line for FSP never goes above the line for PS. In fact, it is well below PS in all the graphs for most of the job size distribution. This property makes FSP attractive, since it provides most of the advantages of SRPT, without a crossover effect. For some job sizes, FSP outperforms SRPT, while for other job sizes, FSP is worse than SRPT. Neither policy dominates the other. As with SRPT, the advantages of FSP over PS are more pronounced for the empirical trace than for the synthetic trace.

To summarize, the SRPT policy is attractive because it optimizes mean response time, but it can create unfairness for some job sizes. The arrival process has a larger relative impact on the PS policy than on the SRPT policy, while the variability of the job size distribution has a larger influence on the SRPT policy than on the PS policy. The FSP policy shows trends similar to SRPT.

The different properties of these scheduling policies, such as their fairness and their sensitivities, motivate our new scheduling policies in the next section.

5 Hybrid SRPT Policies

In this section, we propose two new scheduling policies, each of which is a variant of SRPT. Our intent is to find a scheduling policy with response time (slowdown) properties similar to SRPT, but with fairness properties similar to PS and FSP. We first define our scheduling policies, and then evaluate them, comparing them with SRPT, PS, and FSP.

5.1 K-SRPT

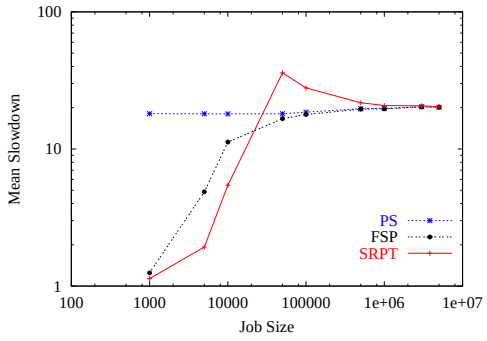
One possible criticism of SRPT is that it dedicates all system resources to a single job at a time, thereby depriving other jobs of service. Our first new scheduling policy generalizes SRPT to provide greater sharing of system resources, but with the same fixed aggregate service rate. We call this policy K-SRPT.

K-SRPT is a multi-threaded version of SRPT that allows up to K jobs (the K smallest RPT ones) to be in service concurrently, as in the PS policy. Additional jobs in the system, if any, must wait in the queue, without receiving service. Two special cases deserve mention. When $K = 1$, the K-SRPT policy is the same as SRPT. When $K \to \infty$, K-SRPT asymptotically converges to PS.

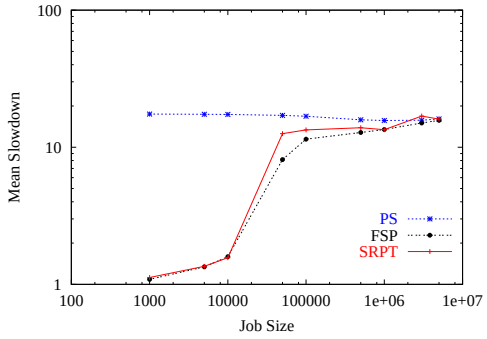
The K-SRPT policy is also preemptive like SRPT. If a new job arrives into the system, and is among the smallest K jobs now in the system, then it preempts the Kth smallest job and begins service immediately.

5.2 T-SRPT

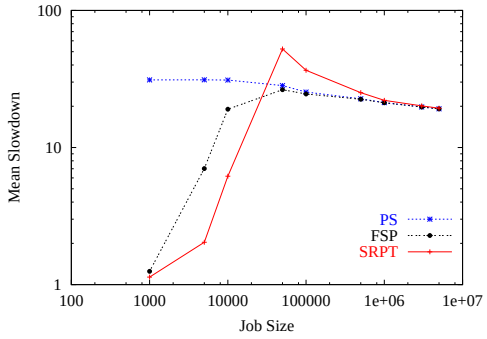
The second new policy is called Threshold-based SRPT, or T-SRPT for short. T-SRPT is a hybrid scheduling policy that combines aspects of SRPT and PS. PS provides fair and equal service to all jobs in the



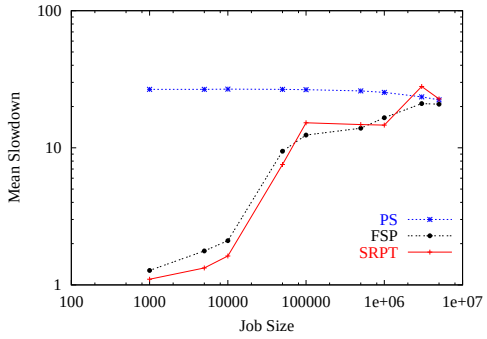
(a) Poisson Arrivals, Exponential Sizes



(b) Poisson Arrivals, Empirical Sizes



(c) Empirical Arrivals, Exponential Sizes



(d) Empirical Arrivals, Empirical Sizes

Figure 2. Slowdown Profile Plots for PS, SRPT, and FSP for Four Different Workloads

```
/* Use the threshold (T) to determine whether
 * the system is "busy" or not. This depends
 * on the number of jobs (J) in the system. */
If J ≤ T
  For each job i = 1, 2, 3, ... J do
    Each job receives service at rate 1/J
  end for i
Else
  The SRPT job receives service at the full rate
```

Figure 3. Algorithmic Description of T-SRPT

system, while SRPT can quickly complete small jobs in the system, reducing overall mean response time.

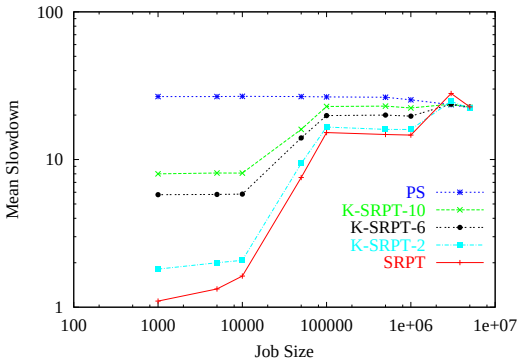
The T-SRPT policy switches back and forth between PS and SRPT depending on the number of jobs currently in the system. If the system is lightly loaded, then it uses PS to provide fair service to all pending jobs. If the system is heavily loaded, such that a backlog builds, then the policy switches to SRPT until the backlog dissipates sufficiently. The threshold T is the boundary between these two modes of operation. Note that the aggregate service rate remains constant; we change only the job scheduling policy.

Figure 3 gives an algorithmic description of T-SRPT. The policy is governed by a single threshold value T, which is a settable parameter. If $T = 1$, the policy always uses SRPT. If $T = \infty$, the policy always uses PS.

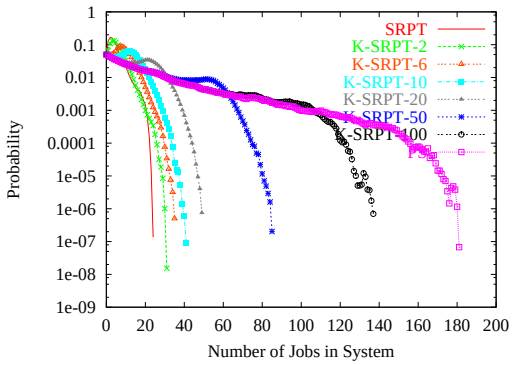
5.3 Results for K-SRPT and T-SRPT

Figure 4 shows the simulation results for our new scheduling policies. All simulation results shown are for the empirical Web workload. The plots in the left-hand column are slowdown profile plots for the different scheduling policies, while the plots in the righthand column show the distribution of the number of jobs in the system. Each graph has multiple lines, for different parameter settings in the respective scheduling policies.

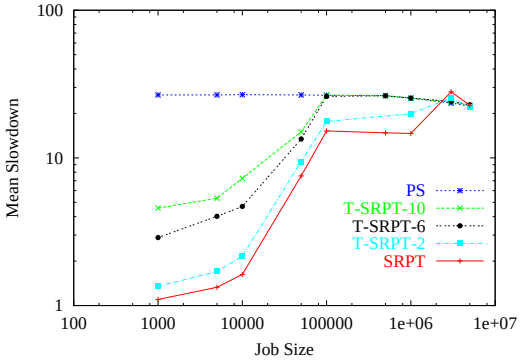
Figure 4(a) shows the slowdown profile plot for K-SRPT, for $K = 2, 6, 10$. For comparison purposes, the plots for SRPT (the lowest line) and PS (the horizontal upper line) are also shown. This graph shows that changing K produces a family of curves falling between SRPT and PS. There is a smooth transition from SRPT to PS as K is increased. Increasing K sacrifices some of the response time advantages of SRPT (as expected), while gaining fairness advantages of PS. The crossover region is reduced and eliminated as K grows larger. By sharing resources amongst K jobs, K-SRPT makes it less likely that a large job is unfairly disadvantaged.



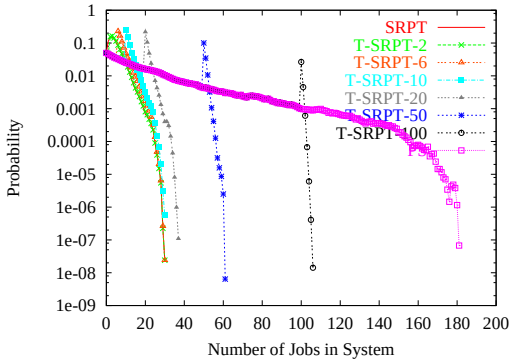
(a) Slowdown Profile Plot for K-SRPT



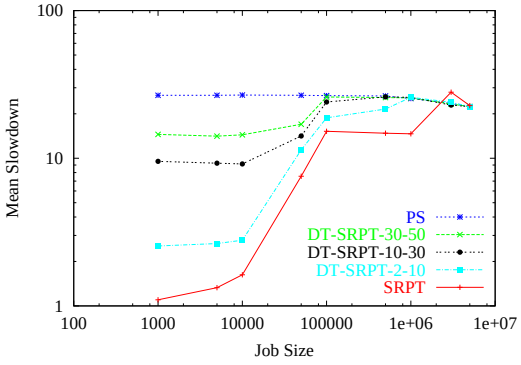
(b) Jobs in System for K-SRPT



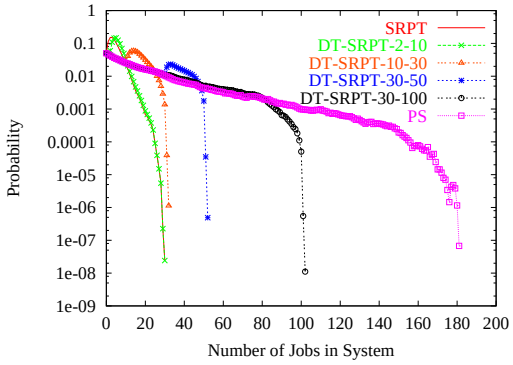
(c) Slowdown Profile Plot for T-SRPT



(d) Jobs in System for T-SRPT



(e) Slowdown Profile Plot for DT-SRPT



(f) Jobs in System for DT-SRPT

Figure 4. Simulation Results for Hybrid SRPT Scheduling Policies (Empirical Trace)

Figure 4(b) illustrates the number of jobs in the system for the K-SRPT policy. The results for the PS policy are indicated by the rightmost line in the graph, with up to 180 jobs in the system at a time. Moving to the left across the graph, the successive lines are for $K = 100$ (at most 140 jobs in the system), $K = 50$ (at most 85 jobs in the system), $K = 20$, and so on. Small values of K limit the tail of this distribution, providing SRPT-like performance.

The simulation results for T-SRPT are shown in Figure 4(c) and Figure 4(d). Figure 4(c) shows the slowdown profile plot for T-SRPT, for $T = 2, 6, 10$. The comparison lines for SRPT and PS are also shown.

Figure 4(c) shows that T-SRPT provides a smooth transition from SRPT to PS as T is increased. This behaviour is as expected. The slowdown performance for small jobs in T-SRPT is slightly worse than for SRPT, but the crossover effect is eliminated as the threshold T is increased.

Comparing K-SRPT in Figure 4(a) and T-SRPT in Figure 4(c) for similar values of K and T yields inconclusive results. T-SRPT is better for smaller job sizes, while K-SRPT is better for larger job sizes. Both policies reduce or eliminate the crossover effect. However, recall that this is a statistical result averaged over many jobs. Neither policy provides the per-job sample path guarantee of FSP [5].

Figure 4(d) provides more information about T-SRPT, by showing the distribution of the number of jobs in the system for different settings of the threshold parameter T. This graph shows that the T-SRPT policy provides tighter control on the tail of the distribution, compared to K-SRPT. For any given setting of T, the plot matches that of the PS policy until the threshold T is reached. The distribution is then effectively truncated at that point. T-SRPT thus provides more precise control of system resource usage (e.g., active TCP connections, number of processes, queue size) than the K-SRPT policy. Interestingly, a spike is created in the distribution near the value T. This spike is induced by the hybrid policy repeatedly switching back and forth between SRPT and PS when the threshold is crossed. This behaviour reflects system instability.

5.4 Double-Threshold SRPT

The inherent instability of the T-SRPT policy at the threshold value T motivates a simple extension of this policy, called Double-Threshold SRPT (DT-SRPT). This policy uses *two* threshold values: a high threshold T_H at which the policy switches from PS to SRPT, and a low threshold T_L at which it switches back from SRPT to PS. Separating T_L and T_H provides greater stability for the hybrid scheduling policy.

Figure 4(e) and Figure 4(f) present the simulation results for DT-SRPT. The slowdown profile plot for DT-SRPT is structurally similar to that of T-SRPT in Figure 4(c), at least for the threshold values considered. However, the marginal distribution plot for the number of jobs in the system for DT-SRPT (Figure 4(f)) is quite different from that for T-SRPT (Figure 4(d)). DT-SRPT provides a smoother transition between scheduler states compared with T-SRPT, while still providing strong control over the tail of the distribution. In fact, DT-SRPT provides greater control of the tail, since the SRPT policy is allowed to dissipate system backlog until T_L is reached.

We further illustrate the advantages of DT-SRPT via SRPT residency time analysis. That is, given that the hybrid policy has just switched from PS to SRPT, how long does it typically stay in the SRPT state before reverting to PS. This time has practical implications on the context switching overhead of T-SRPT.

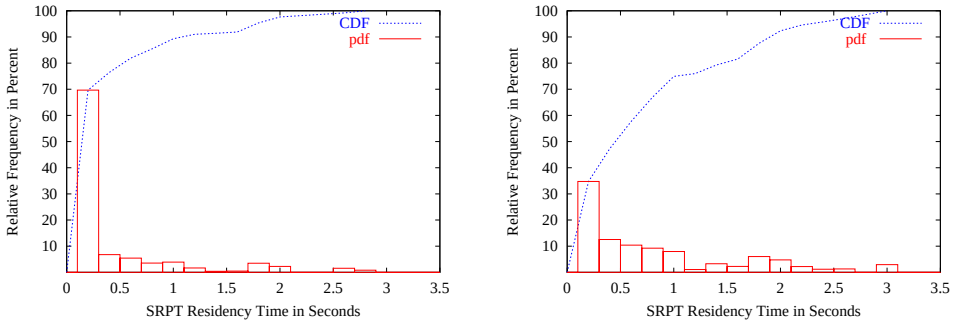
Figure 5 shows the results from SRPT residency time analysis. Figure 5(a) shows the residency time distribution for T-SRPT, with $T = 10$. For this threshold value, there were 240,042 transitions to the SRPT policy during the simulation (and 240,042 transitions back to PS). The T-SRPT policy typically spends only a fraction of a second in the SRPT state per transition. This represents significant overhead for the scheduler.

Figure 5(b) shows results for the DT-SRPT policy, for $T_L = 10$ and $T_H = 30$. For these settings, there were only 1,271 transitions to the SRPT policy during the simulation. Figure 5(b) shows that the scheduler spends more than 1 second in the SRPT state (per transition) about 25% of the time.

Table 3 provides a statistical summary of the simulation results from the hybrid SRPT scheduling policies. DT-SRPT clearly offers greater stability for a practical hybrid scheduling policy.

Table 3. Summary of Simulation Results for Hybrid SRPT Scheduling Policies

Scheduling Policy	Mean Slowdown	SRPT State	State Changes
K-SRPT-2	1.338	N/A	N/A
K-SRPT-10	4.774	N/A	N/A
T-SRPT-2	1.389	74.5%	310,236
T-SRPT-10	3.295	38.5%	480,084
DT-SRPT-2-10	2.505	60.6%	6,554
DT-SRPT-10-30	9.572	22.8%	2,542



(a) T-SRPT ($T = 10$) (b) DT-SRPT ($T_L = 10$, $T_H = 30$)

Figure 5. SRPT Residency Time Analysis for T-SRPT and DT-SRPT

6 Summary and Conclusions

This paper studies the tradeoffs between fairness and responsiveness for Web server scheduling policies. The paper makes three main contributions. First, it offers a direct comparison between the PS, SRPT, and FSP policies using simulation, illustrating slowdown versus job size, as well as sensitivity to the arrival process and job size distribution in empirical and synthetic traces. Second, the paper proposes two novel Web server scheduling policies, each of which is a parameterizable variant of SRPT. The simulation results for these policies show that they provide smooth tradeoffs between SRPT and PS, while also providing control over system resource consumption. Third, the paper identifies a stability problem for the T-SRPT hybrid scheduling policy, illustrating the issue with residency time analysis. The DT-SRPT policy can provide responsiveness, fairness, and stability in Web server scheduling.

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 Trans. on Networking*, Vol. 5, No. 5, pp. 631-645, 1997.

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

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] D. Raz, H. Levy, and B. Avi-Itzhak, "A Resource-Allocation Queueing Fairness Measure", *Proceedings of ACM SIGMETRICS*, New York, NY, pp. 130-141, June 2004.

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

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