Applications of SRPT Scheduling with Inaccurate Information

Dong Lu* Peter A. Dinda* Yi Qiao* Huanyuan Sheng† Fabián E. Bustamante*
*Department of Computer Science
†Department of Industrial Engineering and Management Sciences
Northwestern University
{donglu,pdinda,y-qiao3,h-sheng,fabianb}@northwestern.edu

Abstract

The Shortest Remaining Processing Time (SRPT) scheduling policy was proven, in the 1960s, to yield the smallest mean response time, and recently it was proven its performance gain over Processor Sharing (PS) usually does not come at the expense of large jobs. However, despite the many advantages of SRPT scheduling, it is not widely applied. One important reason for the sporadic application of SRPT scheduling is that accurate job size information is often unavailable. Our previous work addressed the performance and fairness issues of SRPT scheduling when job size information is inaccurate. We found that SRPT (and FSP) scheduling outperforms PS as long as there exists a (rather small) amount of correlation between the estimated job size and the actual job size. In the work we summarize here, we have developed job size estimation techniques to support the application of SRPT to web server and Peer-to-Peer server side scheduling. We have evaluated our techniques with extensive simulation studies and real world implementation and measurement.

1. Introduction

The Shortest Remaining Processing Time (SRPT) scheduling policy is extremely promising because even in a general queuing system (G/G/1), it is provably optimal, leading to smallest possible mean value of occupancy and therefore of delay time [19]. More recent work has shown that the variance of delay time in $M/G/1/SRPT$ queuing systems is lower than FIFO and LIFO [17]. Bansal, et al proved theoretically that the degree of unfairness under SRPT is surprisingly small assuming an M/G/1 queuing model and heavy-tailed job size distribution [3]. Gong, et al further investigated the fairness issues of SRPT through simulation [7] and confirmed the theoretical results regarding the asymptotic convergence of scheduling policies for slowdown [10].

Recently, SRPT [3, 9, 7] has received much attention in the context of connection scheduling in web servers. Harchol-Balter, et al prototyped SRPT scheduling on Apache web server and their evaluation showed the superiority of SRPT over PS [9] in terms of mean response time. To further improve the fairness of SRPT scheduling, Friedman, et al proposed the Fair Sojourn Protocol (FSP) that combines SRPT with PS to trade off fairness with performance [5]. They concluded that FSP is both efficient in a strong sense (similar to SRPT), and fair, in the sense of guaranteeing that it weakly outperforms processor sharing (PS) for every job on any sample path.

All of the previous research on size-based scheduling assumes accurate a priori knowledge of job sizes, which is not obtainable in many cases. Our recent work [14] studied the performance of SRPT and FSP when only inaccurate job size information are available. We found that both SRPT and FSP outperform PS provided a reasonably good job size estimator is available. In particular, the performance of SRPT increases as the correlation between the job size estimates and the actual job sizes increases. If the correlation is too low, PS outperforms both.

In this paper, we summarize our work in applying SRPT scheduling with inaccurate job size information to web server and the server side of peer-to-peer (P2P) systems. We have developed very accurate job size estimators for both domains, making SRPT applicable. Further reports on our work are available elsewhere [14, 13, 18] or under submission.

2. Web server scheduling

The common assumption when applying SRPT scheduling to web servers is that the size of the file to be served is

Effort sponsored by the National Science Foundation under Grants ANI-0093221, ACI-0112891, ANI-0301108, EIA-0130869, and EIA-0224449. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation (NSF).

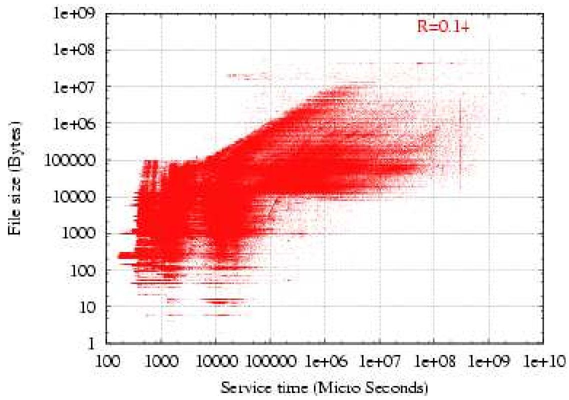

Figure 1. File size versus service time.

a good estimate of the service time of the request. Our measurements suggest that this is not the case, but we have developed a new job size estimator that is sufficiently accurate to preserve much of SRPT's performance.

2.1. Is file size a good estimator of service time?

Size-based SRPT scheduling appeared in digital communication networks research in 1983 [4]. In this context, the service time was taken to be equal to the transmission time of a message, which is proportional to the length of the message stored in the node buffers. A web server serving static requests appears superficially similar in that it transmits files to the client. However, there are differences. First, in the digital communication network context, the work represented by the service time is pushing the bits of the message onto the wire, while for the web server context, the work involves end-to-end cooperation along an entire shared heterogeneous path. Although most transfers are likely to be dominated by the bottleneck bandwidth in the path and the latency of the path, there are multiple possible bottlenecks along the path and they can vary with time due to packet losses and congestion. Second, the disk(s), memory system(s), and CPU(s) of the web server and the client are also potential bottlenecks. These complexities suggest that the service time of a request may not be proportional or even well correlated with the size of the file it serves.

We use the correlation coefficient between file size and service time (the commonly used Pierson's R) [1] to decide on the effectiveness of the estimator.

We modified Apache's log module so that it records the *response time* each request with microsecond granularity. When deployed, we measure the load conditions on the disk, network, and CPU. If these resources have zero or near-zero load, then our response time measurement is also a measurement of the service time.

File Size	R
$X < 30$ KB	0.0616
$30 \leq X < 500$ KB	0.1121
$X > 500$ KB	0.1033

Figure 2. R depends on file size.

We deployed the module on our department-level web site and validated that the load was near zero. We collected data from September 15, 2003 to October 19, 2003. This trace includes approximately 1.5 million HTTP requests, among which less than 2% are dynamic PHP requests that collectively took less than 1% of the total service time recorded. $>$ 98% of our requests and $>$ 99% of the service time in the trace are for static pages. Hence, our web sever is dominated by static web content. Others claim that static content dominates web traffic [11, 9] and thus our results are comparable to theirs. The requests originated from 110 "/8" IP networks, 7220 "/16" IP networks and 31250 "/24" IP networks spread over the world. We claim that this server is typical. However, our conclusions are also supported by other measured traces and generated traces.

Given the provenance of the trace, we can now use it to answer our question. Figure 1 is a log-log scatter plot of file size versus service time. Visually, we can see hardly any correlation between file size and service time. File transfer times vary over several orders of magnitude with same file size. *Over the entire 1.5 million requests in the trace, we find that R is a very weak 0.14.* R varies slightly with file size, as can be seen in Figure 2.

2.2. Domain-based service time estimator

Given that request file size and service time are weakly correlated, and that the performance of size-based scheduling policies are strongly dependent on the degree of this correlation [14], a natural question is whether there is a better service time (job size) estimator than file size, one whose estimates are more strongly correlated with actual service time. Such an estimator must also be lightweight, requiring little work per request.

Our domain-based estimator relies on the Internet being statistically stable over periods of time, particularly from the point of view of the web server. Fortunately, there is significant evidence that this is the case. Previous research showed that the Internet not only shows routing stability [16], but also spatial locality and temporal locality in end-to-end TCP throughput [2, 15, 12].

Although the Internet, web servers, and clients form a highly dynamic system, this stability suggests that previous web requests (the web server's access log) are a rich history which can be used to better estimate the service time of a new request. We assume that after processing a request we know (1) its file size, (2) the actual service time, and (3) the IP address of the client. Collecting this information is simple and efficient. We use a history of such requests, combined with the file size and IP address of the current request to determine the likely service time of the current request, with the goal of achieving a higher correlation between the estimated service time and the actual service time that is higher than the correlation between file size and actual service time. The correlation R must exceed a threshold in order for SRPT to perform better than PS, and as R increases, the performance of SRPT increases.

Consider a *domain*, a neighborhood in the network topology. The broad use of Classless Inter-Domain Routing [6] (CIDR) implies that routes from machines in the domain to a server outside the domain will share many hops. Similarly, the routes from the server to different machines in the domain will also have considerable overlap. This also means that the routes will be likely to share the same bottleneck network link and therefore have similar throughput to/from the server. Smaller domains have more sharing.

The aggregation of CIDR is along a hierarchy of increasingly larger networks and is reflected in IP addresses. The first k bits of an IP address gives the network of which the address is a part, the first $k-1$ bits give the broader network that contains the first network, and so on. We exploit this hierarchy in domain-based scheduling, the algorithm of which is given below.

1 Use the high order k bits of the client IP address to classify the clients into 2^k domains, where the k bits are treated as the domain address.

2 Aggregate past requests to estimate the service rate (or representative bandwidth) for each domain. This can be done with several estimators, but our experiments show that the estimator $S_R = \frac{F_s}{S_t}$ performs the best. Here S_R is the representative service rate, F_s is the sum of the requested file sizes from the domain, and S_t is the sum of the service times for these requests. Notice that updating this estimate after a request has been processed is trivial: simply add the request's file size and service time to F_s and S_t, respectively (two reads, two adds, two writes). For each domain, we store F_s and S_t. An array of these pairs is kept, indexed by the domain address. The total state size is 2^{k+1} floating point numbers.

3 For each incoming client request, the web server first extracts the domain address, indexes the array and computes S_R for the domain. It then estimates the request's service time as $T_{estimate} = \frac{f_s}{S_R}$, where f_s is the request file size. The estimator requires a logical shift, two reads, a division, and a multiply. For a request from a heretofore unobserved domain, which occurs exactly once per domain, we simply use file size as the estimate.

4 Apply a size-based scheduling policy such as SRPT using the estimated service times.

As we might expect, as domains become smaller (k gets larger), predictive performance increases, at the cost of memory to store the state. SRPT outperforms PS significantly with the support of the domain-based service time estimator. A detailed performance evaluation can be found in our technical report [13].

Statistics	Service Time	Served Chunk Size	Requested Chunk Size
Service Time	1.0000	0.7023	0.2833
Served Chunk Size	0.7023	1.0000	0.2339
Requested Chunk Size	0.2833	0.2339	1.0000

Figure 3. Correlation coefficients between service time, served chunk size and requested chunk size.

3. P2P server side scheduling

In the context of peer-to-peer file sharing, research efforts have focused on routing, search, incentives, and a few other topics. However, as far as we are aware, our ongoing work [18] is alone in looking at the server side scheduling problem. The goal of this work is to schedule download requests at the server side of P2P systems so as to minimize the average response time experienced by P2P users.

3.1. Job sizes in a P2P server

We studied the workload of P2P servers and compared several alternative scheduling policies for server side scheduling. To do so, we collected a large set of object request traces from Gnutella, one of the most popular P2P file-sharing networks.

Interestingly, for P2P scheduling, there are four different possible definitions for job size: *full object size*, *requested data chunk size*, *served data chunk size*, and *job service time*. While the full object size is usually very large, most requested data chunks are small, covering only a small fraction of the whole object. More importantly, there is usually also a significant difference between the requested data chunk size and the actual served data chunk size. Figure 3 shows clearly that there is some degree of correlation between service time and requested data chunk size. The correlation between served chunk size and service time is even stronger, but neither can be known until the job finishes.

3.2. Preliminary results

Since a typical P2P download request is for a specific chunk of the whole object, we could use the requested chunk size as a rough estimate of service time, and as the metric for SRPT scheduling. The much stronger correlation between served chunk size and service time, on the other hand, indicates that served chunk size can be a better estimate for service time.

We explored how SRPT performs when using requested chunk size (CS) and served chunk size (SS) as the schedul-

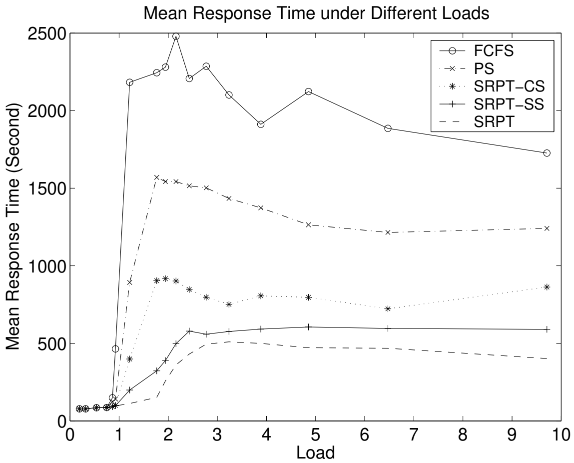

Figure 4. Mean Response time for different scheduling policies under varying load.

ing metric. For comparison purposes, we also studied the scheduling performance for ideal SRPT. The three scheduling policies are denoted as SRPT-CS, SRPT-SS, and SRPT, respectively. Notice that *SRPT-CS can be directly implemented with current tools*, while SPRT-SS would require an accurate estimator. The performance of First-Come-First-Serve (FCFS) and Processor-Sharing (PS) scheduling policies are also studied.

Figure 4 gives the mean response time of the five scheduling policies handling all requests for a P2P server, with the varying system load. The advantages of the three SRPT-based policies over PS and FCFS are clear, especially when the load is close to or above 1. This regime is important because previous work has shown that P2P servers often become overloaded [8]. Even with only weak correlation between requested chunk size and service time, SRPT-CS still gives us a much shorter response time than FCFS or PS. *This suggests deploying SRPT-CS would significantly improve the user experience of Gnutella and tools like it.* The gaps between SRPT-CS, SRPT-SS, and ideal SRPT reveal that there is opportunity to further improve performance by developing accurate predictors of served chunk size or service time.

4. Conclusions and future work

We have applied SRPT scheduling with estimated job sizes in web server scheduling and P2P server side scheduling. For web server scheduling, we found that file size is not a good estimator for its service time and developed a domain-based service time estimator that dramatically enhances the performance of SRPT scheduling. For P2P server side scheduling, we have demonstrated the utility of using SRPT with a very simple estimator and we are currently working on more sophisticated estimators.

References

[1] ALLEN, A. O. *Probability, statistics, and queueing theory with computer science applications*. Academic press, Inc., 1990.

[2] BALAKRISHNAN, H., SESHAN, S., STEMM, M., AND KATZ, R. H. Analyzing Stability in Wide-Area Network Performance. In *Proceedings of ACM SIGMETRICS* (June 1997).

[3] BANSAL, N., AND HARCHOL-BALTER, M. Analysis of SRPT scheduling: investigating unfairness. In *Proceedings of SIGMETRICS/Performance* (2001), pp. 279–290.

[4] BUX, W. Analysis of a local-area bus system with controlled access. *IEEE Transactions on Computers 32*, 8 (1983), 760–763.

[5] FRIEDMAN, E. J., AND HENDERSON, S. G. Fairness and efficiency in web server protocols. In *Proceedings of SIGMETRICS/Performance* (2003).

[6] FULLER, V., LI, T., YU, J., AND VARADHAN, K. (rfc1519) Classless Inter-Domain Routing (CIDR): an address assignment and aggregation strategy, September 1993.

[7] GONG, M., AND WILLIAMSON, C. Quantifying the properties of srpt scheduling. In *Proceedings of IEEE MASCOTS* (2003).

[8] GUMMADI, K. P., DUNN, R. J., SAROIU, S., GRIBBLE, S. D., LEVY, H. M., AND ZAHORJAN, J. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. 19th ACM SOSP* (2003).

[9] HARCHOL-BALTER, M., SCHROEDER, B., BANSAL, N., AND AGRAWAL, M. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems (TOCS) 21*, 2 (May 2003).

[10] HARCHOL-BALTER, M., SIGMAN, K., AND WIERMAN, A. Asymptotic convergence of scheduling policies with respect to slowdown. *Performance Evaluation 49*, 1/4 (2002).

[11] KRISHNAMURTHY, B., AND REXFORD, J. *Web Protocols and Practice: HTTP1.1, Networking Protocols, Caching, and Traffic Measurements*. Addison-Wesley, 2001.

[12] LU, D., QIAO, Y., DINDA, P., AND BUSTAMANTE, F. Characterizing and predicting tcp throughput on the wide area network. Tech. Rep. NWU-CS-04-34, Northwestern University, Computer Science Department, April 2004.

[13] LU, D., SHENG, H., AND DINDA, P. Effects and implications of file size/service time correlation on web server scheduling policies. Tech. Rep. NWU-CS-04-33, Northwestern University, Computer Science Department, April 2004.

[14] LU, D., SHENG, H., AND DINDA, P. Size-based scheduling policies with inaccurate scheduling information. In *Proceedings of IEEE MASCOTS* (2004).

[15] MYERS, A., DINDA, P. A., AND ZHANG, H. Performance characteristics of mirror servers on the internet. In *Proceedings of IEEE INFOCOM* (1999), pp. 304–312.

[16] PAXSON, V. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking 5*, 5 (1997), 601–615.

[17] PERERA, R. The variance of delay time in queueing system M/G/1 with optimal strategy SRPT. *Archiv fur Elektronik und Uebertragungstechnik 47*, 2 (1993), 110–114.

[18] QIAO, Y., LU, D., BUSTAMANTE, F., AND DINDA, P. Looking at the server side of peer-to-peer systems. In *7th Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR 2004)* (2004).

[19] SCHRAGE, L. E., AND MILLER, L. W. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research 14* (1966), 670–684.