Report Number: WUCSE-2006-1b

2006-01-01

Fair Efficiency, or Low Average Delay without Starvation

Christoph Jechlitschek and Sergey Gorinksky

Elastic applications are primarily interested in minimal delay achievable for their messages under current network load. In this paper, we investigate how to transmit such messages over a bottleneck link efficiently and fairly.

2006-1

90day_evaluation

Authors: Someone1 and someone

Corresponding Author: jpw4@cec.wustl.edu

Abstract: contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *
FROM tblFileInfo i INNER JOIN tblFiles f on i.FileID = f.FileID
WHERE FileInfoID = 456, contents for abstract for the page
SELECT *

Type of Report: Other

Fair Efficiency, or Low Average Delay without Starvation

Christoph Jechlitschek and Sergey Gorinsky

Technical Report WUCSE-2007-16
Department of Computer Science and Engineering, Washington University in St. Louis
One Brookings Drive, St. Louis, MO 63130-4899, USA
{*chrisj,gorinsky*}*@arl.wustl.edu*

February 28, 2007

***Abstract*— Elastic applications are primarily interested in minimal delay achievable for their messages under current network load. In this paper, we investigate how to transmit such messages over a bottleneck link efficiently and fairly. While SRPT (Shortest Remaining Processing Time) is an optimally efficient algorithm that minimizes average delay of messages, large messages might starve under SRPT in heavy load conditions. PS (Processor Sharing) and ViFi (Virtual Finish Time First) are fair but yield higher average delays than under SRPT. We explore the class of fair algorithms further and prove that no online algorithm in this class is optimally efficient. Then, we derive a fair algorithm SFS (Shortest Fair Sojourn) and report extensive experimental evidence that SFS is consistently more efficient than PS and ViFi during either temporal overload or steady-state operation, with the largest benefits when average load is around the bottleneck link capacity. Furthermore, average delay under the fair SFS remains close to the minimum attained under the unfair SRPT.**

I. Introduction

Cross-layer optimization encompasses a variety of recent efforts to overcome inefficiencies that isolation between layers imposes on the traditional network design [1], [2]. In this paper, we explore whether and by how much explicit accounting for application needs improves efficiency of network capacity allocation. While applications in general might require highly diverse network services, our study focuses on elastic applications, which are often seen as the most harmonious fit for the Internet layered architecture.

Elastic applications are interested in minimal delay achievable for their messages under current network load. If one reduces the problem of network capacity allocation to scheduling a single bottleneck link, Shortest Remaining Processing Time (SRPT) – which schedules messages preemptively in the order of their remaining transmission delays – is an optimally efficient algorithm because SRPT minimizes average delay of messages [3]. However, the optimal efficiency of SRPT comes at the expense of potential unfairness: in some settings with heavy load, SRPT starves large messages by delaying them without bound [4].

Processor Sharing (PS) is an alternative algorithm that instantaneously allocates equal shares of the bottleneck capacity to all pending messages [5]. Consequently, expected delay of a message under PS is proportional to the message size. Also, since PS does not rely on knowledge of message sizes, PS lends itself nicely to implementation in layered network designs. Due to the above reasons, PS has become a traditional ideal in network capacity allocation. Although packet-switching networks do not support instantaneous sharing of a link, a lot of research has been conducted on packet transmission algorithms that approximate the PS ideal. Developed packet-grained approximations comprise fair queuing at routers – such as Weighted Fair Queuing (WFQ) [6] or Deficit Round Robin (DRR) [7] – and fair end-to-end congestion control exemplified by Transmission Control Protocol (TCP) [8].

While SRPT is unfair, PS achieves fairness by sacrificing efficiency: average delay of messages under PS is significantly higher. Recent studies reveal remarkable existence of algorithms that have it both ways and combine fairness with SRPT-like efficiency. Virtual Finish Time First (ViFi) is a specific efficient representative of the fair algorithmic class where no message is delayed longer than under PS [9]. ViFi schedules messages preemptively in the order of their finish times under PS and is independently proposed as Fair Sojourn Protocol (FSP) in the context of web servers [10]. Significant reductions in average delay under ViFi versus PS are substantiated both experimentally and analytically [9]–[11].

This paper sheds more light on the class of fair algorithms for network capacity allocation. First, we show that the fair class does not contain an optimally efficient online algorithm. Then, we develop Shortest Fair Sojourn (SFS), a fair algorithm with even lower average delay than under ViFi in most settings. Our extensive simulations over a wide range of network load illustrate efficiency and fairness properties of SFS, ViFi, PS, and SRPT. In particular, we demonstrate that average delay under SFS versus ViFi is consistently lower over the whole range of the experiments.

The rest of the paper is structured as follows. Section II clarifies our model, metrics, and terminology. Section III rules out existence of an optimally efficient algorithm in the fair class. This section also presents SFS and proves its fairness. Section IV reports the experimental comparison of SFS, ViFi, PS, and SRPT. Finally, Section V sums up and discusses our findings.

II. Model, Terminology, and Metrics

We define a *message* as an atomic data unit meaningful for an application. Messages are supplied for network transfer in their entirety, e.g., as supported by the DOT (Data-Oriented Transfer) service interface [12]. *Delay* of a message is time passed from the message arrival until the whole message reaches its destination. Related studies refer to delay under other names such as transfer time, response time, flow time, or sojourn time. *Transmission delay* of a message represents its communication needs and equals $\frac{S}{C}$, where S is the message size, and C is the capacity of the network bottleneck link shared with all the other messages. Transmission delay is also known as processing time, e.g., as reflected in the name of SRPT. We assume that the communications utilize the whole bottleneck capacity and have negligible propagation and node processing delays. These assumptions are reasonable for bulk-data transfers that are of primary interest for us (in more general settings, the extra delays could have been modeled more precisely as constants, and such model adjustment would not affect the qualitative conclusions of our analysis). Then, besides transmission delay, the only other component of message delay is due to waiting for the bottleneck link to become available.

Network load is characterized by arrival times and transmission delays of messages. Network service is represented by an algorithm that allocates the bottleneck link capacity to pending messages. In particular, we are interested in *online algorithms* that have no access to information about future messages. Capacity allocation enjoys ideal flexibility that allows both instantaneous link sharing and instantaneous transmission preemption.

Since PS has become synonymous with fairness in network resource allocation, we rely on delays of individual messages under PS as a basis for defining the fair algorithmic class:

Definition 1: The *fairness constraint* dictates that no message should finish later than under PS. *Starvation* refers to a scenario where the fairness constraint is violated. An algorithm for capacity allocation is *fair* if and only if no starvation occurs under the algorithm for any network load.

To quantify fairness of an algorithm to a particular message, we introduce a metric of *starvation stretch*:

Definition 2: Starvation stretch $s_{\text{x}}(m)$ of message m under algorithm X is the ratio of message delay $d_{\text{x}}(m)$ under algorithm X to message delay $d_{\text{PS}}(m)$ under PS:

$$s_{\text{x}}(m) = \frac{d_{\text{x}}(m)}{d_{\text{PS}}(m)}. \tag{1}$$

Note that algorithm X is deemed unfair if there exists network load where $s_{\text{x}}(m) > 1$ for at least one message m.

Also note an implicit assumption that network capacity is allocated among messages. We strongly believe that fairness of capacity allocation should be defined with respect to real-world entities, rather than messages or packet flows as in traditional networking. However, since the important "among what" aspect is orthogonal to our main contributions and requires a separate thorough treatment, we do not explore it further in this paper.

Message	Arrival time	Transmission delay
1 through 9	0	10 each
10	0	14
11	90	10
12	90	20

Fig. 1. Network load from the proof of Theorem 1.

To quantify efficiency of network capacity allocation under algorithm X, we measure average delay D_{x} for all n messages in imposed network load:

$$D_{\text{x}} = \frac{\sum_{m=1}^{n} d_{\text{x}}(m)}{n}. \tag{2}$$

Because SRPT is an optimally efficient algorithm if fairness concerns are ignored, we use average delay under SRPT as a baseline for assessing efficiency of fair algorithms:

Definition 3: Average letup L_{x} under algorithm X is the ratio of average delay D_{x} under algorithm X to average delay D_{SRPT} under SRPT:

$$L_{\text{x}} = \frac{D_{\text{x}}}{D_{\text{SRPT}}}. \tag{3}$$

Although a fair algorithm is not always able to match the ideal efficiency of the unfair SRPT, consistent closeness of average letup L_{x} to 1 is an indicator that fair algorithm X is highly efficient.

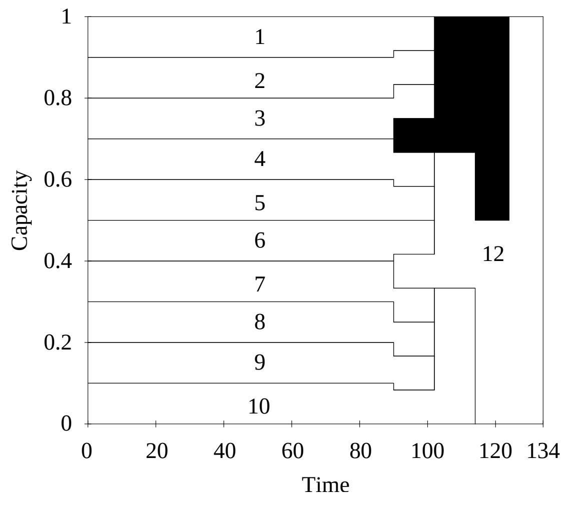
III. Improving on ViFi

Both PS and ViFi are fair algorithms but ViFi provides significantly lower average delay [9]. Is ViFi the most efficient among fair algorithms? Or if it is not, does the fair class contain another online algorithm that minimizes average delay? The following theorem gives the negative answer to both above questions.

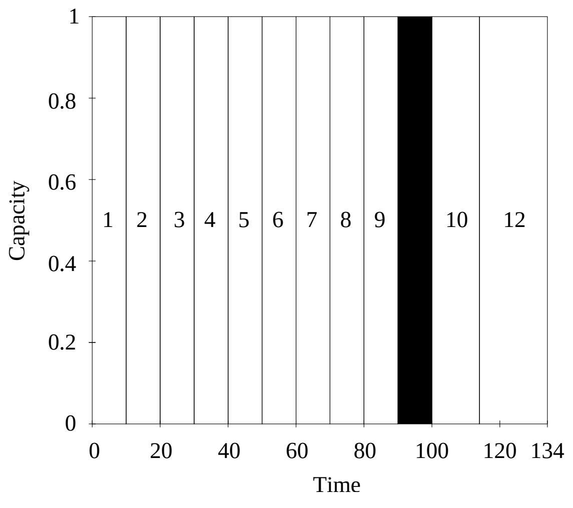
Theorem 1 (No Optimal Online Algorithm): No online algorithm minimizes average delay without starvation.

Proof: Consider a scenario with 12 messages that have arrival times and transmission delays as specified in Figure 1. Since SRPT does not violate the fairness constraint, it is optimal to transmit the messages in an SRPT order: any permutation of 1 through 9 followed by 11, 10, and 12. Figure 2b depicts the optimal schedule, which provides average delay $\frac{618}{12}$. Note that average delay under ViFi, or any other algorithm that transmits message 10 before messages 11 and 12, is higher and equals $\frac{622}{12}$.

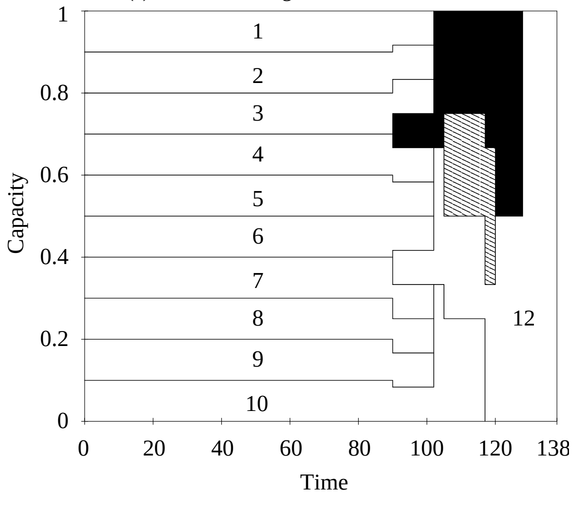
Suppose that message 13 with transmission delay 4 arrives at time 105. By time 105 in the above optimal schedule, messages 1 through 9 and 11 have finished, and message 10 is being transmitted since time 100. Suspending message 10 to transfer message 13 would violate the fairness constraint because the resumed message 10 would complete at time 118 whereas Figure 2c shows that message 10 finishes under PS at time 117. Hence, message 10 has to finish before messages



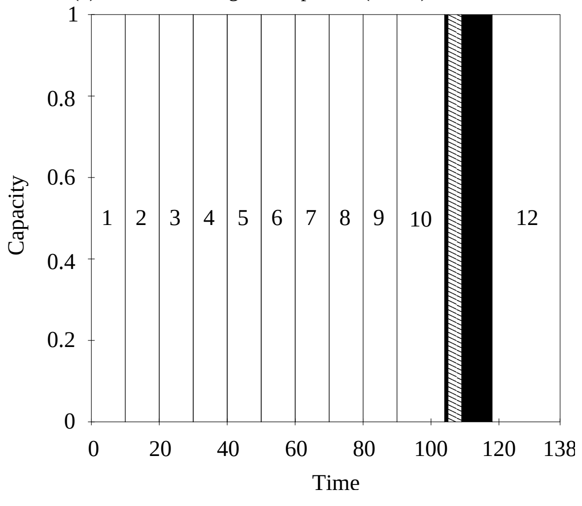
(a) without message 13: PS schedule



(b) without message 13: optimal (SRPT) schedule



(c) with message 13: PS schedule



(d) with message 13: optimal (ViFi) schedule

Fig. 2. No optimally effi cient online algorithm in the fair class: messages 11 and 13 are denoted with solid black and stripes respectively.

13 and 12 are transmitted. The resulting schedule provides average delay $\frac{635}{13}$. However, smaller average delay $\frac{634}{13}$ would be achieved if the 13 messages were transmitted in a ViFi order: any permutation of 1 through 9 followed by 10, 11 (suspended at time 105 to transfer message 13), 13, 11 (the rest of it), and 12. Figure 2d depicts this optimal ViFi schedule.

Thus, optimality of transmitting message 11 (rather than message 10) at time 90 depends on whether message 13 arrives at time 105. There is no optimal online algorithm that minimizes average delay without starvation. ■

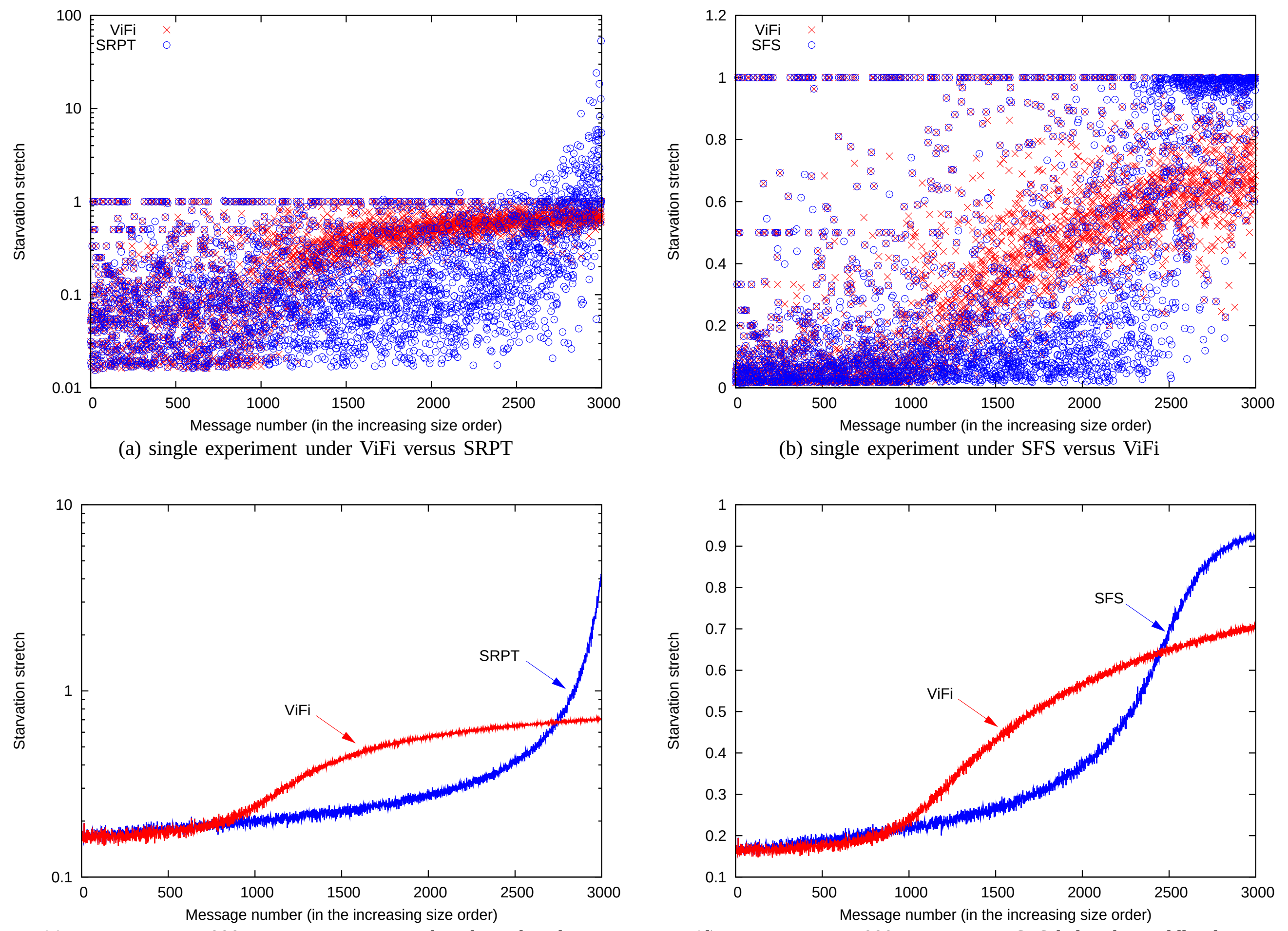
While Theorem 1 precludes existence of a fair online algorithm that minimizes average delay, some algorithms might outperform ViFi with respect to average delay in most settings. To design such algorithms, we rely on two insights. First, selecting a message with shorter remaining transmission delay yields lower average delay. Second, while ViFi always transmits messages in the order of their PS finish times, deviation from the PS order to transmit a message with shorter remaining transmission delay does not necessarily violate the fairness constraint.

Shortest Fair Sojourn (SFS) is an algorithm based on the above insights. Let M be a pending message with the shortest remaining transmission delay. SFS checks whether transmitting M first and then scheduling the other pending messages in a ViFi order does not lead to starvation. If the schedule does not violate the fairness constraint, SFS transmits M. Otherwise, SFS transmits a message with the smallest PS finish time. The following theorem establishes that SFS is fair.

Theorem 2 (Fairness of SFS): SFS is a fair algorithm.

Proof: Messages might starve only because arrival of a new message N increases their SFS finish times past their PS finish times. If N or another message is chosen for transmission despite not having the smallest PS finish time (i.e., the chosen message has the shortest remaining transmission delay), then the updated schedule avoids starvation by SFS definition. Otherwise, SFS schedules N and the other pending messages in a ViFi order. The updated schedule is such that:

- Message N completes by its PS finish time due to fairness of ViFi [9].



(a) single experiment under ViFi versus SRPT

(b) single experiment under SFS versus ViFi

(c) averages over 1,000 experiments: ViFi makes the rich richer, and the poor get richer too

(d) averages over 1,000 experiments: SFS helps the middle class

Fig. 3. Dependence on message sizes at 95% load.

- Messages with smaller PS finish times than N's are scheduled before N and hence complete even before their prior PS finish times.
- Messages with the same or larger PS finish times than N's have their PS finish times postponed by the transmission delay of N and hence complete under SFS at or before their extended PS finish times.

In all scenarios, no message starves. SFS is fair. ∎

We also consider SFS+, a computationally more extensive variant of SFS. SFS+ sorts pending messages in the increasing order of their remaining transmission delays and selects the first message F (up to a message with the smallest PS finish time) such that transmitting F and scheduling the other pending messages in a ViFi order does not violate the fairness constraint. Our simulations show that SFS and SFS+ yield similar average delays. Moreover, SFS+ average delay exhibits a surprising tendency of being slightly higher. Hence, experiments reported in our next section compare ViFi, PS, and SRPT with SFS only.

IV. Performance Evaluation

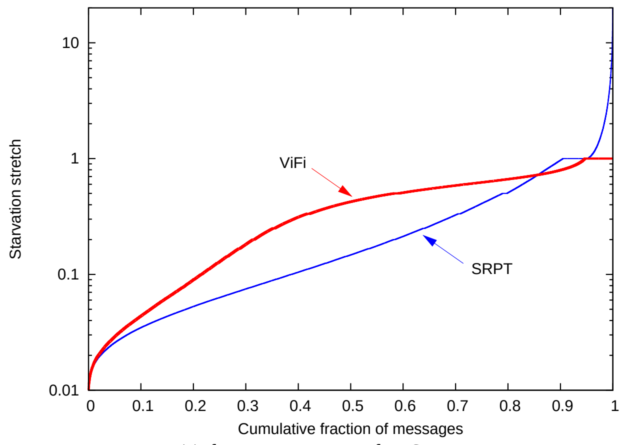
A. *Experimental methodology*

We simulate transmission of 3,000 messages over a link of capacity $C = 10$ Tbps, with full link utilization and no data loss. Our choices for the link capacity and traffic are dictated by our desire to model broadband networks of the future. Message sizes and arrival times are drawn from random distributions. For each set of the traffic settings, we repeat the experiment under SRPT, PS, ViFi, and SFS.

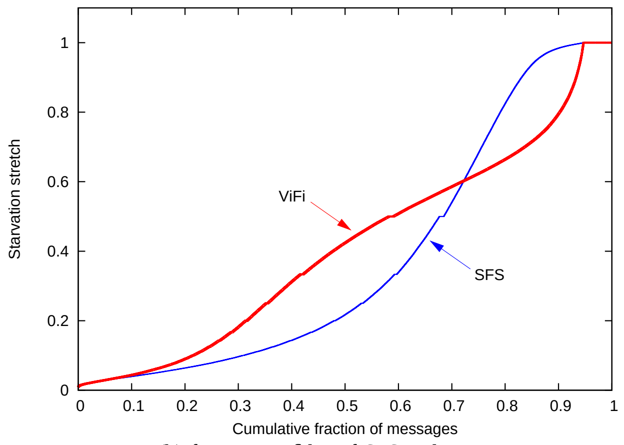
To characterize intensity of the traffic, we define a notion of *load* l as

$$l = \frac{m}{C \cdot t} \tag{4}$$

where m denotes the average message size, and t is the average message interarrival time. Note that since the number of messages is finite, all message delays remain finite even with $l > 100\%$. This feature of our experimental setup enables us to compare the evaluated algorithms under long-term overload conditions, which is impossible with analytical techniques that target only steady-state algorithmic behaviors.



(a) fair ViFi versus unfair SRPT



(b) fairness profi les of SFS and ViFi

Fig. 4. Fairness of ViFi and SFS versus unfairness of SRPT: cumulative distributions of starvation stretches over 100 experiments at 95% load.

We experiment with both uniform and Pareto distributions of message sizes. Unless explicitly stated otherwise, we report results for message sizes that are uniformly distributed between 100 GB and 100 TB. For Pareto-sized messages, our experiments confirm the surprising finding by Bansal and Harchol-Balter [13] that even under heavy loads, SRPT starves only few messages and does not inflate starvation stretches much beyond 1. In these settings, the provenly fair ViFi and SFS still provide far superior efficiency in comparison to PS. To illustrate the efficiency gain, we also plot average letups under PS, ViFi, and SFS in scenarios where message sizes are drawn from the Pareto distribution with index 1.5 and minimum message size of 500 GB. In all our experiments, messages arrive according to a Poisson process with such an average rate that yields a desired value of load.

The code and running instructions for all the reported simulations are available at our web site [14].

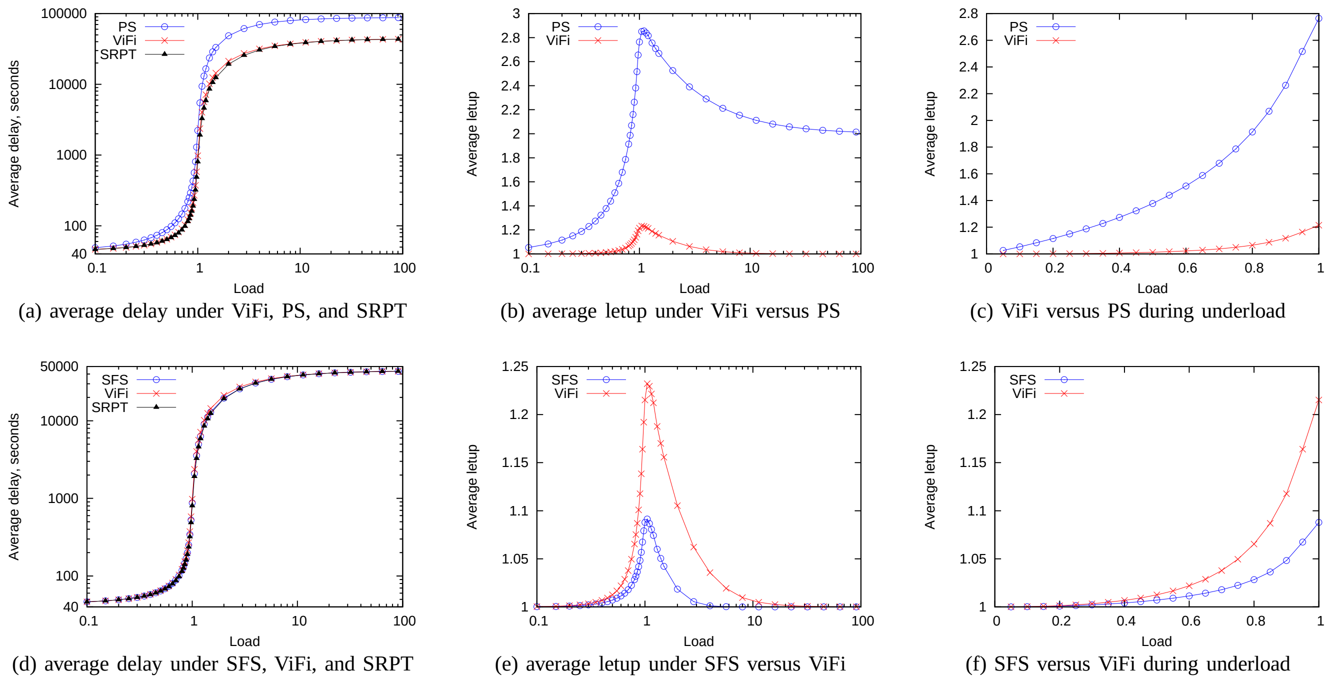
B. Dependence on message sizes

We start by examining starvation stretches of all 3,000 messages in a single experiment at 95% load. Figure 3a plots the starvation stretches under SRPT and ViFi in the increasing order of message sizes. Each message is depicted on the plot with an individual point. Under both algorithms, a small fraction of messages across the whole spectrum of message sizes has starvation stretch 1. These messages finish at exactly the same times as under PS because they conclude a traffic burst by emptying the queue upon their completion (under both SRPT or ViFi and PS). The graph also clearly illustrates the unfairness of SRPT. Small and even midsize messages benefit significantly from SRPT, which delivers them up to 50 times faster than under PS. However, some large messages starve. For example, delay for the least lucky message under SRPT is about 50 times larger than under PS.

For the fair ViFi, Figure 3a shows that 800 smallest messages enjoy similarly low starvation stretches as under SRPT. To explain the similarity, we observe that a small message is likely to possess both the shortest remaining transmission delay and earliest PS finish time among pending messages. For larger messages, the ViFi profile becomes different. Starvation stretches of midsize messages rise significantly closer to 1 than under SRPT. On the other hand, the increase enables ViFi to complete all large messages by their PS finish times.

As Figure 3b illustrates, SFS also schedules small messages similarly to ViFi: respective plotted points often coincide. The reason for the similarity is the same as for SRPT versus ViFi. Again, SFS and ViFi differ in their treatment of midsize and large message. A dense cluster of points around starvation stretch 1 for large messages under SFS indicates that SFS reduces delays for midsize messages by postponing large messages almost as long as possible without causing starvation. In addition to the across-the-spectrum line at starvation stretch 1, Figure 3b also reveals sparser but still discernible rows of points with starvation stretches $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$. The rows correspond to messages that arrive and finish while 1, 2, or 3 other messages remain pending (under both SFS or ViFi and PS).

To expose the discussed trends more clearly, we repeat the experiment 1,000 times and average the 1,000 obtained sets of starvation stretches sorted in the increasing order of message sizes. Figure 3c shows that SRPT substantially decreases delays of small and midsize messages but the largest messages typically starve. Under ViFi, not only small messages (the rich) benefit from abandoning PS but also the largest messages (the poor) have average starvation stretch about 0.7. Hence, ViFi improves upon PS across the board by reducing delays for all classes of messages: rich, middle, and poor! Figure 3d illustrates strategic differences between SFS and ViFi. By keeping starvation stretches of large messages closer to 1, SFS helps the middle class of midsize messages to enjoy significantly lower delays than under ViFi.



(a) average delay under ViFi, PS, and SRPT (b) average letup under ViFi versus PS (c) ViFi versus PS during underload

(d) average delay under SFS, ViFi, and SRPT (e) average letup under SFS versus ViFi (f) SFS versus ViFi during underload

Fig. 5. Efficiency of PS, ViFi, and SFS versus SRPT.

C. *Cumulative distributions of starvation stretches*

The average starvation stretches reported in Figures 3c and 3d blur fates of individual messages. Hence, we also plot cumulative distributions of all 300,000 starvation stretches in 100 instances of our experiment. Comparing ViFi with SRPT, Figure 4a shows that while starvation stretches up to the 85-th percentile are higher under the fair ViFi, the top 5% of starvation stretches under the unfair SRPT exceed 1, i.e., belong to starving messages. Figure 4b compares cumulative distributions of starvation stretches under SFS versus ViFi. The main divide lies around 73%. Up to the 73-rd percentile, starvation stretches are lower under SFS. Under either SFS or ViFi, the top 5% of starvation stretches equal 1. Between the 73-rd and 95-th percentiles, ViFi yields smaller starvation stretches. Similarly to Figures 3a and 3b, the lines in Figures 4a and 4b contain horizontal segments at starvation stretches 1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and $\frac{1}{5}$. These flat segments reflect messages that arrive and finish (under both SRPT or ViFi or SFS and PS) while 0, 1, 2, 3, or 4 other messages remain pending.

D. *Results on efficiency*

To evaluate efficiency of the algorithms, we conduct our experiment for various values of load l. We repeat the experiment 1,000 times for each examined load $l \le 120\%$, i.e., including all examined instances of underload, but generally less for overloads of $l > 120\%$. Figures 5a and 5d illustrate an intuitive expectation that average delays under SRPT, PS, ViFi, and SFS grow as load increases. After load hits and surpasses 100%, the delays remain finite and even decelerate their growth because the number of messages in every experiment is finite. For the extreme of "infinite" load when all 3,000 messages arrive simultaneously, the average delays are analytically expressed by Gorinsky and Rao [9]. In particular, PS yields the following average delay in a single experiment with simultaneous message arrivals:

$$D^{\infty}_{\text{PS}} = \frac{\sum_{k=1}^{n} (2(n-k)+1)m_k}{nC} \tag{5}$$

where m_k is the size of the k-th smallest message, $n = 3{,}000$ is the number of messages, and $C = 10$ Tbps is the link capacity. When the messages arrive simultaneously, SRPT, SFS, and ViFi produce an identical transmission schedule for the experiment and achieve the same average delay [9]:

$$D^{\infty}_{\text{SRPT}} = D^{\infty}_{\text{SFS}} = D^{\infty}_{\text{ViFi}} = \frac{\sum_{k=1}^{n} (n-k+1)m_k}{nC}. \tag{6}$$

For the considered uniform distribution of message sizes, we derive the expected average delay under PS as:

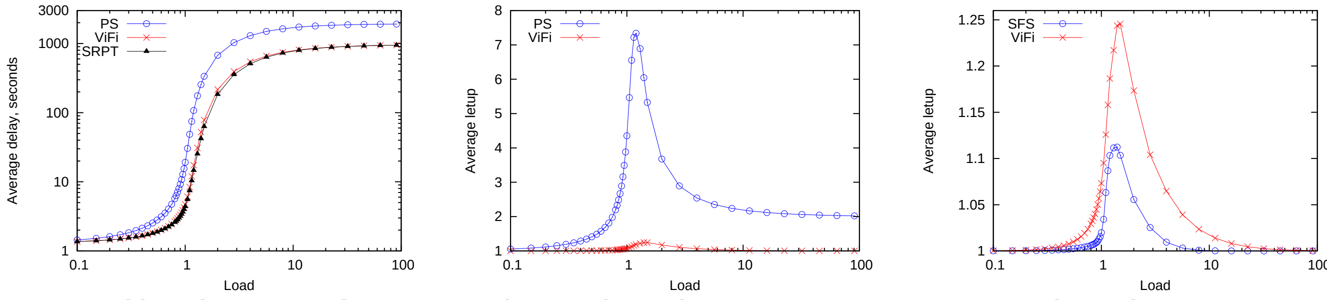
$$D^{\infty}_{\text{PS}} = \frac{(4n+1)m_{\text{min}} + (2n-1)m_{\text{max}}}{6C} \approx 88{,}118 \text{ seconds}$$

where $m_{\text{min}} = 100$ GB and $m_{\text{max}} = 100$ TB are respectively minimum and maximum message sizes in the distribution. The expected average delay under SRPT, SFS, and ViFi becomes:

$$D^{\infty}_{\text{SRPT}} = D^{\infty}_{\text{SFS}} = D^{\infty}_{\text{ViFi}} = \frac{(n+1)(2m_{\text{min}} + m_{\text{max}})}{6C} \approx 44{,}081 \text{ sec.}$$

Figures 5a and 5d confirm that experimental average delays converge asymptotically to the above analytical predictions.

Figures 5b and 5e plot average letups under PS, ViFi, and SFS. All three letups peak around $l = 100\%$. At this load



(a) average delay under ViFi, PS, and SRPT (b) average letup under ViFi versus PS (c) average letup under SFS versus ViFi

Fig. 6. Efficiency of PS, ViFi, and SFS with Pareto-sized messages.

where the arrival rate matches the link capacity, PS, ViFi, and SFS have respectively 2.8 times, 22%, and 9% larger average delays than under SRPT. Asymptotically, the average letup under PS converges to:

$$L^{\infty}_{\text{PS}} = \frac{(4n+1)m_{\text{min}} + (2n-1)m_{\text{max}}}{(n+1)(2m_{\text{min}} + m_{\text{max}})} \approx 2$$

while SFS and ViFi converge to the optimal efficiency:

$$L^{\infty}_{\text{SFS}} = L^{\infty}_{\text{ViFi}} = 1.$$

In general, SFS provides SRPT-like efficiency with consistently lower average delays than even under ViFi.

While our results for $l > 100\%$ offer interesting insights into behavior of the algorithms in long-term overload conditions, Figures 5c and 5f focus on underload scenarios $l < 100\%$ which are the most relevant for steady-state operation. Again, SFS consistently outperforms ViFi. For example, when load equals 80%, average delays under PS, ViFi, and SFS are respectively 2 times, 7%, and only 3% worse then the minimum attained under the unfair SRPT.

Finally, we explore efficiency of PS, ViFi, and SFS for the Pareto distribution of message sizes. As Figures 6a, 6b, and 6c illustrate, Pareto-sized messages reap even greater benefits from abandoning PS in favor of the efficient representatives of the fair class. Average delays under PS, ViFi, and SFS peak around 7.3 times, 25%, and 11% above the minimum provided by the unfair SRPT. Once again, SFS consistently supports the highest efficiency among the examined fair algorithms.

V. Conclusion

In this paper, we studied a class of fair algorithms for network capacity allocation where no message finishes later than under PS. In addition to PS, the fair class includes ViFi and newly proposed SFS (Shortest Fair Sojourn). We proved that no online algorithm in the fair class is optimally efficient with respect to average delay of messages. Nevertheless, our extensive experiments demonstrated that SFS is consistently more efficient than PS and ViFi during either temporal overload or steady-state operation, with the largest benefits when average load is around the bottleneck link capacity. Furthermore, average delay under the fair SFS remains close to the minimum attained under the unfair SRPT. Our simulations revealed that SFS and ViFi gain their significant efficiency improvements over PS across the whole spectrum of message sizes, including large messages but primarily due to dramatic delay reductions for small messages. To outperform ViFi, SFS decreases delays for midsize messages by postponing large messages almost as long as possible without causing starvation.

We conducted our investigation within a simple model that assumed a single bottleneck link and allowed both instantaneous link sharing and instantaneous transmission preemption. In our future work, we will build upon the discovered insights to design efficient and fair algorithms for transmitting messages of elastic applications over networks with multiple bottleneck links. To deal with multiple distributed bottlenecks, computational complexity, and other envisioned challenges, we will learn from previous research on message-grained transmission over packet-switching networks [15], [16].

References

[1] M. Conti, E. Gregori, and G. Turi, "A Cross-Layer Optimization of Gnutella for Mobile Ad hoc Networks," in *Proceedings ACM MobiHoc 2005*, May 2005.

[2] J. Wang, L. Li, S. H. Low, and J. C. Doyle, "Cross-Layer Optimization in TCP/IP Networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 582–595, June 2005.

[3] L. E. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline," *Operations Research*, vol. 16, no. 3, pp. 687–690, May-June 1968.

[4] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan, "Flow and Stretch Metrics for Scheduling Continuous Job Streams," in *Proceedings ACM-SIAM SODA 1998*, January 1998.

[5] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, 1987.

[6] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proceedings ACM SIGCOMM 1989*, September 1989.

[7] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings ACM SIGCOMM 1995*, September 1995.

[8] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings ACM SIGCOMM 1988*, August 1988.

[9] S. Gorinsky and N. S. V. Rao, "Dedicated Channels as an Optimal Network Support for Effective Transfer of Massive Data," in *Proceedings High-Speed Networking (HSN 2006)*, April 2006.

[10] E. J. Friedman and S. G. Henderson, "Fairness and Efficiency in Web Server Protocols," in *Proceedings ACM SIGMETRICS 2003*, June 2003.

[11] A. Wierman and M. Harchol-Balter, "Bounds on a Fair Policy with Near Optimal Performance," Carnegie Mellon University, Tech. Rep. CMU-CS-03-198, November 2003.

[12] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An Architecture for Internet Data Transfer," in *Proceedings Networked Systems Design and Implementation (NSDI 2006)*, May 2006.

[13] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness," in *Proceedings ACM SIGMETRICS 2001*, June 2001.

[14] C. Jechlitschek and S. Gorinsky, "Simulation Suite for Comparative Studies of PS, SRPT, ViFi and SFS," February 2007, http://www.arl.wustl.edu/~chrisj/sfs.

[15] E. Modiano, "Scheduling Packet Transmissions in a Multi-hop Packet Switched Network Based on Message Length," in *Proceedings ICCCN 1997*, September 1997.

[16] L. Cherkasova, "Scheduling Strategy to Improve Response Time for Web Applications," in *Proceedings High-Performance Computing and Networking (HPCN Europe 1998)*.