Differentiation Between Short and Long TCP Flows: Predictability of the Response Time

Konstantin Avrachenkov[†] , Urtzi Ayesta[†,*], Patrick Brown[*], Eeva Nyberg[‡]

[†]INRIA Sophia Antipolis, {k.avrachenkov, uayesta}@sophia.inria.fr
[*]France Telecom R&D, {Patrick.Brown,Urtzi.Ayesta}@francetelecom.com
[‡] Helsinki University of Technology, Eeva.Nyberg@netlab.hut.fi.

***Abstract*— Internet measurements show that a small number of large TCP flows are responsible for the largest amount of data transferred, whereas most of the TCP sessions are made up of few packets. Several authors have invoked this property to suggest the use of scheduling algorithms which favor short jobs, such as LAS (Least Attained Service), to differentiate between short and long TCP flows.**

We propose a packet level stateless, threshold based scheduling mechanism for TCP flows, RuN2C. We describe an implementation of this mechanism which has the advantage of being TCP compatible and progressively deployable. We compare the behavior of RuN2C with LAS based mechanisms through analytical models and simulations. As an analytical model, we use a two level priority Processor Sharing $PS + PS$. In the $PS+PS$ system, a connection is classified as high or low priority depending on the amount of service it has obtained. We show that $PS + PS$ reduces the mean response time in comparison with standard Processor Sharing when the hazard rate of the file size distribution is decreasing. By simulations we study the impact of RuN2C on extreme values of response times and the mean number of connections in the system.

Both simulations and analytical results show that RuN2C has a very beneficial effect on the delay of short flows, while treating large flows as the current TCP implementation does. In contrast, we find that LAS based mechanisms can lead to pathological behavior in extreme cases.

Keywords. TCP/IP, $M/G/1$, Processor Sharing, LAS, Response time, Simulations, Queueing theory.

I. Introduction

We study the differentiation between short and long flows in a TCP/IP network. There are several reasons to favor short flows. The reasons are based on user ergonomics, on the design of TCP congestion control, on the file size distribution and on queueing theory. Most user interactions with a network or with applications running across a network consist either entirely of short interchanges or, of short interchanges followed by a longer transfer. If the sum of transfer times were constant, from an ergonomics point of view, it would seem preferable to diminish the transfer times of short transactions at the expense of transfer times of longer transactions, as they do not require as much user attention as short transactions.

TCP applications react to congestion and losses by reducing their window sizes either with a certain fluidity through fast retransmit procedures or after a timeout. For short connections with small window sizes, a loss is often detected only after a timeout and possibly after all data has been sent to the network. As a result, timeouts on short connections are not very effective in reducing the overall traffic and stabilizing the network [1]. A loss occurrence for a short TCP transfer may thus increase the transfer time manifold while, on the other hand, a reduction in the latency of the order of one second would be a significant improvement [2] for short transactions.

From a queueing theory point of view, it has been shown that choosing an appropriate scheduling policy may significantly improve the performance of the system. One of the classical results of queuing theory says that the Shortest Remaining Processing Time (SRPT) policy is optimal for the network as it is able to reduce the overall mean latency of the flows [3]. This is only applicable, if there is use of a priori knowledge of the flow sizes, which obviously is not the case in the current TCP/IP architecture. When the size jobs is not known, LAS has been proposed as a good approximation of $SRPT$. Yashkov showed in [4], (see also [5]) that LAS scheduling policy is optimal with respect to the average time in the system among all work conserving disciplines that do not take advantage of precise knowledge of the job lengths, when the service time distribution has a decreasing hazard rate (DHR). The same result is presented by Righter and Shantikumar in [6]. Harchol et al. [7] showed that for distributions with DHR LAS effectively reduces the average time in the system with respect to PS.

From the traffic point of view, it is known that Internet flow size distributions exhibit heavy tailed behavior (see, e.g. [8], [9], [10]) and can often be modeled by a distribution with a decreasing hazard rate, e.g. Pareto distribution. Thus most TCP sessions, e.g. interactive sessions, are of small size but a small amount of large flows, e.g. from data applications, are responsible for the largest amount of transferred data. As a consequence, if scheduling policies that favor short connections were to be implemented in the Internet, the average time in the system could be reduced.

Lastly, the perceived quality is more prone to high variations in transfer times of short flows than of long flows and, as we will argue, in the current TCP/IP network short transactions have wild variations in transfer times, in part due to the conservative value of the retransmission timer [11].

In several recent works [12], [13], [14], [15], [16], [17], the authors address the differentiation between short and long flows in Internet networks. In [12] the authors suggest two ap-

0-7803-8355-9/04/$20.00 ©2004 IEEE.

proaches based solely on simulation studies. The first approach is application based and it is proposed in the framework of Differentiated Services (DiffServ), with Assured Forwarding (AF) and RED with In and Out (RIO). This approach requires a non trivial choice of the numerous AF and RIO parameters. The second approach is TCP state based, using each connection's window size and relying on the compliance of the end hosts. Furthermore, this approach requires tuning of weighted round robin (WRR) parameters, which again is neither evident nor robust.

In [13], [14], [15] the authors propose a two class based architecture to provide better service to short TCP flows. At the edge router, state information is kept for active flows. Packets are marked with high priority if the current length of the flow is below some threshold and inside the network service differentiation is performed by RIO routers or WRR scheduling. They present the gain obtained on mean response times through simulations. The results presented show reasonable gain in the average performance, but with no discussion on the worst case performance or on the variance of the performance. In [14], [15], the authors also discuss the analytical modeling of their approach, but are only able to give approximate numerical results based on the use of the Kleinrock's conservation law [3].

In [16] and ensuing work [17] the authors study the LAS (or FB) scheduling policy on the flow level and what the LAS policy would produce in the context of a TCP network if packets from TCP flows were sorted in decreasing order of attained service.

The above mentioned papers have two main drawbacks. The first drawback is that all the previous works rely only on one metric, the mean conditional response time for given flow size. We notice that dramatically different mechanisms such as Last Come First Serve ($LCFS$) and PS give identical mean conditional response times [3]. In contrast, we judge the effectiveness of size-based scheduling mechanisms for TCP flows not only on mean conditional response times but also on extreme values of response times. We also analyze the system stability by measuring the number of ongoing connections. Through these performance measures, we are able to give a thorough and improved picture of the benefits of our scheme on the predictability of the performance of the flows and on the stability of the network.

The second drawback of previous works, is that the proposed implementation mechanisms and simulation studies rely on buffer architectures requiring the tuning of many parameters, e.g. RIO and WRR. In contrast, we propose the use of simple priority queues with Drop Tail that has the advantage of being robust and scalable.

We propose a novel mechanism, RuN2C, as an implementation of a threshold based procedure, which does not necessitate state management. This mechanism has the advantage of being implementable in different parts of a network: in access networks on the transmission side, in backbone networks (although authors have argued that such mechanisms are less useful in backbone networks) or more interestingly on the reception side of an access network. Also, the mechanism does not require prior agreements between ISPs to be used across networks and is robust to small variations, e.g., orders of 10 packets in the value of the threshold.

We will use the following terms. We will call RuN (Running Number differentiation mechanism) an implementation of LAS over a TCP/IP network, where TCP packets are given priority according to the rank of their first data byte in the connection's data stream. We will call RuN2C (Running Number 2 Class differentiation mechanism) an implementation over a TCP/IP network of a two priority class processor sharing ($PS + PS$) scheduling mechanism in which packets are classified as high or low priority according to the comparison of the packet's first data byte rank in the connection's data stream with a given threshold.

We derive asymptotical results based on analytical models of the proposed mechanisms, assuming Drop Tail routers, to show that threshold based mechanisms produce close to standard TCP transfer times for large transfers while LAS may discriminate severely against large transfers. We prove analytically that when the service time distributions have decreasing hazard rates the overall mean response time in a system employing $PS+PS$ scheduling is always smaller than the mean response time in a PS system. Furthermore, we show numerically that by an appropriate choice of the threshold, the overall mean response time in the system can be chosen close to the minimum response time in a LAS system.

The novel threshold based mechanism, RuN2C, is then compared to LAS based mechanisms and standard Drop Tail scheduling through simulations. We show that RuN2C produces an important improvement of performance with respect to all cited metrics for short flows, while treating large flows as the current TCP implementation does.

The rest of the paper is organized as follows. Section II discusses the theoretical justifications for the RuN2C mechanism in comparison to standard TCP over Drop Tail queues and to LAS. Section III provides details for a possible implementation of RuN2C in a TCP/IP network. Section IV provides a more exhaustive study of RuN and RuN2C via simulations. In particular, we study the variability of TCP transfers' response times. The paper is concluded in section V.

II. Algorithms and Mathematical Models

In this section we analyze two packet level scheduling policies for the differentiation between short and long TCP flows. We study these scheduling policies using PS-type flow level models. In particular, we are interested in calculating the mean response time $\overline{T}(x)$ for jobs with service time x. In order to evaluate what impact that giving priority to short connections has on large flows, we study the asymptotics of $\overline{T}(x)$ as $x \to \infty$. The notation $E[\overline{T}]$ is used to denote the overall mean response time. The other metrics of interest are the throughput, $\Theta(X) = x/T(x)$, and the slowdown, $S(x) = T(x)/x$. From the user perception, the mean response time and its variance is an appropriate metric for the HTTP

type web traffic and the average throughput is an appropriate metric for the FTP type traffic. We study these metrics in more detail in Section IV.

Assuming a Poisson flow arrival process, a small Bandwidth Delay Product (BDP), and comparable Round Trip Times (RTTs), the current realization of the TCP/IP network is well represented by the $M/G/1-PS$ queuing model [8], [18]. The performance metrics for the $M/G/1-PS$ model with the capacity scaled to one is given by [3]

$$\overline{T}^{PS}(x) = \frac{x}{1-\rho}, \tag{1}$$

$$\overline{\Theta}^{PS}(x) = 1-\rho, \tag{2}$$

$$\overline{S}^{PS}(x) = \frac{1}{(1-\rho)}. \tag{3}$$

Based on these performance metrics, the Processor Sharing policy is fair in the sense that it gives a constant throughput to all flows regardless of their size. Thus the mean slowdown of the flow is also independent of the flow size.

A. RuN Scheduling Policy

We first analyze the LAS flow level scheduling policy and its packet level implementation RuN. For the LAS implementation one should know the running number of the packet, i.e., one should know if the packet is the first, third, tenth, etc. packet of a flow. The RuN packet level scheduling policy consists in serving the packets arriving to the router buffer in ascending order according to their running number. Hence, the abbreviation for this implementation scheme is RuN. On the flow level, the LAS or Foreground Background (FB_∞) scheduling policy has been thoroughly studied (see e.g. [3])[1].

Let us consider a flow that requires a total service time of x. Let $F(x)$ be the distribution function of the flow service times. Next we denote as $\overline{X_x^n}$ the n-th moment of the truncated distribution at x. Namely,

$$\overline{X_x^n} = \int_0^x y^n dF(y) + (x)^n (1-F(x)).$$

The utilization factor for the truncated distribution is $\rho_x = \lambda \overline{X_x^1}$. This value represents the virtual load a customer of size x sees in the system. From [3] we know that the average response time conditioned on the flow size is

$$\overline{T}^{LAS}(x) = \frac{\overline{W}(x)+x}{1-\rho_x}, \tag{4}$$

with

$$\overline{W}(x) = \frac{\lambda \overline{X_x^2}}{2(1-\rho_x)}. \tag{5}$$

The LAS scheduling policy is optimal with respect to the overall mean response time among the scheduling disciplines that do not use any information about the remaining service time of connections when the file size distribution has a decreasing hazard rate [4], [6]. In particular we have $E[\overline{T}^{LAS}] \leq E[\overline{T}^{PS}]$, even though some large flows suffer (see Figure 2). Furthermore, if the flow size distribution has finite mean and variance, then, as shown in [22],

$$\lim_{x\to\infty} \overline{S}^{LAS}(x) = \lim_{x\to\infty} \overline{S}^{PS}(x) = \frac{1}{1-\rho}.$$

Specifically, consider the Pareto distribution, $F(x) = 1-(k/x)^\alpha$, which is an acceptable approximation for the file size distribution for the current Internet state. Clearly, its hazard rate,

$$\mu(x) = \frac{F'(x)}{1-F(x)} = \frac{\alpha}{x},$$

is a decreasing function, and thus $E[\overline{T}^{LAS}] \leq E[\overline{T}^{PS}]$.

Let us analyze the asymptotics for $\overline{T}^{LAS}(x)$ as x goes to infinity. From expression (4) one immediately obtains that

$$\overline{T}^{LAS}(x) = \frac{1}{1-\rho}x + \frac{\lambda \overline{X^2}}{2(1-\rho)^2} + o(1).$$

We note that if the variance $\overline{X^2}$ is large, the mean conditional response times for the middle size flows in the case of LAS can be significantly larger than in the case of PS. Moreover, if the variance is infinite ($1<\alpha<2$), the asymptotics has the following form

$$\overline{T}^{LAS}(x) = \frac{1}{1-\rho}x + \frac{\lambda k^\alpha}{(1-\rho)^2(2-\alpha)}x^{2-\alpha} + o(x^{2-\alpha}).$$

There is no asymptote in this case, even though the limit $\lim_{x\to\infty} \overline{S}^{LAS}(x)$ exists. This implies that the performance of LAS deviates increasingly from PS performance with the increase of the file size.

Finally, we note that the LAS scheduling policy can be very unfair. Consider the case when many flows of the same size arrive subsequently in such a manner that a new flow enters the system just before the previous flow is about to be finished. In this scenario, all these flows will leave the system at the same time under the LAS policy. There might be a huge difference between the response times for the first and the last flows. On contrary, the PS scheduling discipline will treat all flows fairly in this scenario.

The above remarks and the difficulty in implementing the RuN mechanism prompt us to look for some scheduling policy which combine good properties of both LAS and PS. An example of such a policy is proposed in the next section.

B. RuN2C Scheduling Policy

RuN2C is a threshold based scheduling policy that blends the good features of both LAS and PS. On the flow level, this policy serves flows in the first queue as long as they have received an amount of service less than a given threshold. They then move to the second queue, which is serviced only if the first queue is empty. Inside the queues flows are serviced using the PS discipline. This policy corresponds to the two level PS scheduling policy $M/G/1-PS+PS$ introduced in [3]. On the packet level, we consider a drop tail queue

[1]For the sake of analytical tractability we use here the model with an infinitesimal size of the service quanta given to each user. But the results can be extended to the case of finite service quanta [19], [20], [21].

with a threshold based two class priority mechanism, hence the abbreviation RuN2C for the packet level mapping. Upon a packet arrival in a router, the running number is inferred and is compared to a given threshold value. The packet is marked to either a high priority with small running number or a low priority with high running number. The packets with a number smaller than the threshold (class 1 packets) will be serviced before the packets with a number larger than the threshold (class 2 packets). A possible implementation is presented in Section III.

Let th be the value of the threshold. Given a flow of size x where $x > th$, the first th packets of a flow will have high priority, while the rest $x-th$ packets will be considered as low priority packets.[2] For those flows with a size equal or smaller than th, the system behaves as a pure PS system where the service time distribution is truncated at th. Hence the mean response time is given by

$$\overline{T}^{PS+PS}(x) = \frac{x}{1-\rho_{th}}, \tag{6}$$

for $x \in [0, th]$. The average number of flows served during a busy period is $1/(1-\rho_{th})$. For flows with size $x \in (th, \infty)$ the mean response time conditional on the flow size is given by

$$\overline{T}^{PS+PS}(x) = \frac{\overline{W}(th) + th + \alpha(x-th)}{1-\rho_{th}}. \tag{7}$$

The expression consists of the delay due to the time spent in the first high priority queue, where the flow is serviced up to the threshold th, $(\overline{W}(th)+th)/(1-\rho_{th})$, and the time spent in the lower priority queue $\alpha(x-th)/(1-\rho_{th})$. $\overline{W}(th)$ is given by (5).

The only unknown expression is $\alpha(x)$. It is the virtual time spent in the lower priority queue and to solve it one can consider a queuing system with bulk arrivals to the lower priority queue after a busy period in the high priority queue. This approach results in the following integral equation [3] for $\alpha'(x) = \mathrm{d}\alpha(x)/\mathrm{d}x$

$$\begin{aligned}\alpha'(x) &= \lambda\overline{n}\int_0^\infty \alpha'(y)B(x+y)dy \\ &\quad +\lambda\overline{n}\int_0^x \alpha'(y)B(x-y)dy \\ &\quad +bB(x)+1,\end{aligned} \tag{8}$$

where

$$\overline{n} = \frac{1-F(th)}{1-\rho_{th}}$$

is the mean fraction of flows that reach the low priority queue after a busy period of the high priority queue,

$$b = 2\lambda(1-F(th))\frac{\overline{W}(th)+th}{1-\rho_{th}}$$

[2]We remind that here we consider the normalized service time.

is the average number of flows that arrive to the low priority queue conditioning that at least one does. $B(x)$ is the complementary truncated distribution given by

$$B(x) = \frac{1-F(th+x)}{1-F(th)},$$

which for the exponential case, due to the memoryless property, reduces to the form given in [3], $B(x) = 1-F(x)$.

The integral equation (8) can be solved by using either the finite approximation of the Riemann sum or fixed point iterations. Note that though in [15] the authors use the same approach to model their system, and derive a similar integral equation, they are not able to solve the resulting integral equation, even numerically.

Theorem 1 describes the limiting behavior of the slowdown for large file sizes in the case of $M/G/1-PS+PS$ scheduling policy.

Theorem 1: Let the service time distribution have a finite mean. Then the slowdown for the $PS+PS$ scheduling policy has a limit as the flow size goes to infinity

$$\lim_{x\to\infty}\overline{S}^{PS+PS}(x) = \lim_{x\to\infty}\overline{S}^{PS}(x) = \frac{1}{1-\rho}.$$

The proof of the theorem is postponed to Appendix A. Furthermore, it has been recently shown that the mean conditional response time $\overline{T}^{PS+PS}(x)$ has an asymptote with the slope $1/(1-\rho)$ and with finite bias independent of the second moment of the service time distribution [23].

Based on this result, we note that the advantage of the $PS+PS$ scheduling discipline against LAS is that the absolute difference between $PS+PS$ and PS transfer times remains bounded, even in the case of distributions with infinite variance. As a consequence, very large flows should not see a difference between RuN2C and the current TCP/IP implementation.

Furthermore, for service time distributions with decreasing hazard rate, the average performance, in terms of response times, of $PS+PS$ is better than PS. This a particular result of more general results that will appear in [24].

Theorem 2: Let the service time distribution have a finite mean and decreasing hazard rate. Then the mean overall response time for $PS+PS$ and PS satisfy

$$E[\overline{T}^{PS+PS}] \le E[\overline{T}^{PS}]$$

The proof of the theorem is postponed to the Appendix B.

We study numerically the value of the mean overall response time, $E[\overline{T}^{PS+PS}]$. In Figure 1 we depict the value of the average response time as a function of the threshold for a bounded Pareto flow size distribution with parameters $BP(k,p,\alpha) = BP(13, 5000, 1.1)$ and mean 64 packets. An optimal choice of the threshold in the $PS+PS$ scheme gives nearly the same gain in overall performance as in the case of LAS, without the drawbacks of servicing very long flows unfairly. This result is even more striking if we keep in mind that LAS is optimal with respect to the average time in the

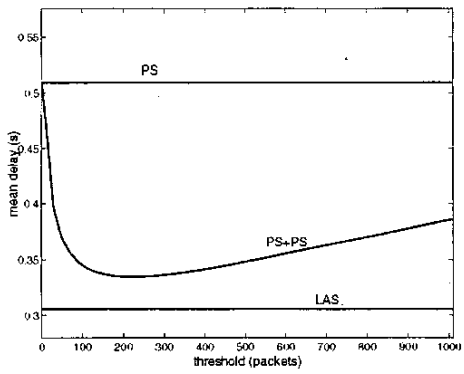

Fig. 1. Optimal value of the threshold for $E[\overline{T}^{PS+PS}]$ for a heavy tailed flow size distrinution.

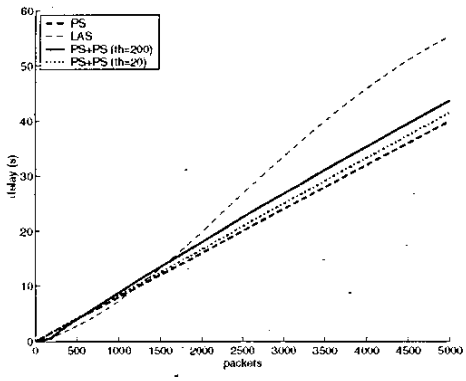

Fig. 2. Difference in delay as a function of flow size for a heavy tailed flow size distribution.

system, when the hazard rate of the service distribution is decreasing, among all the work conserving disciplines that do not take advantage of precise knowledge of the job lengths [4] and [6]. Of course, we recognize that the optimal theoretical value of the threshold may be different from the value which gives the best performance for a real implementation. For a Pareto distribution, with maximum flow size of 5000 packets, the optimal threshold is around 200 packets. In the context of TCP it is sufficient to set up the threshold parameter to a much smaller value (for example 10) in order to guarantee that a TCP connection will recover from a loss by fast retransmit rather than by a costly timeout, see Section III.

Interestingly, even though the benefit that LAS and $PS+PS$ provide to short flows are comparable, $PS+PS$ provokes a smaller degradation of the performance for large flows than LAS. Figure 2 shows the delay for flows for the three different scheduling policies, as a function of their size. The flow size distribution is bounded Pareto with parameters $BP(13, 5000, 1.1)$ and mean 64 packets. We observe that even though large flows do not suffer much with $PS+PS$, the average time in the system is reduced significantly (see Figure 1).

The analytical flow level models presented in this section can be used to model the proposed packet level implementations, as long as the transient effect of TCP congestion control, namely the initial slow start phase is taken into account. Thus, assuming that the TCP maximum window size is the bottleneck, not the BDP of the link and looking at the cycle for every RTT and assuming that an application sends one window, w, worth of packets one packet at a time during one RTT, we have

$$\overline{T}^{SS}(x) = \begin{cases} (\lfloor \log_2 x \rfloor + 1)RTT, & x \leq 2^{\lceil \log_2 w \rceil + 1} \\ \overline{T}_{SS}(x-1) + RTT/w, & \text{otherwise.} \end{cases}$$

Adding the slow start model to the scheduling delay models of this section have a good correspondence with the simulation results of Section IV. Due to limited available space, we only consider the simulation results, but conclude that the delay of short and long TCP flows can be modeled by two main components: slow start and queueing.

III. Implementation of RuN2C for TCP/IP

As explained previously, RuN2C is a threshold based two class differentiation mechanism. The packet level implementation requires maintaining a two class priority queue and knowledge of the value of the threshold parameter between high and low priority queues.

In order to propose a stateless implementation, routers must be able to classify packets from a TCP connection without taking into account previous events. To this end we propose that routers infer the served amount of bytes by only looking at the current TCP sequence number[3]. TCP sequence numbers are incremented from one segment to the other by the number of bytes in the packet's workload and the initial sequence number is picked at random when the TCP connection is established.

The principal retained is to have TCP sequence numbers start from a set of Possible Initial Numbers PIN_i where $i \in \{1, \ldots, 2^R\}$ equally spaced in the sequence number field ranging from 0 to $2^{32}-1$. These numbers should be spaced not too far to allow for the initial sequence number of a TCP connection to be picked sufficiently at random[4]. They must be spaced far enough to reduce the probability (or the occurrence rate) of running over to the next PIN.

Let th be the value in bytes of the threshold, packets for which the sequence number is between PIN_i and $PIN_i + th$ will be classified as priority packets in contrast to packets for which the sequence number lies between $PIN_i + th$ and PIN_{i+1}, where PIN_{i+1} is the next possible initial number (see Figure 3).

With this structure the sequence number expressed in binary code is divided into three parts (see Figure 3). The $R = 32 - (L+TH)$ most significant bits are picked at random, providing 2^R different PIN values. The next L bits and the following TH bits, where $TH = log_2 th$, are set to zero when the TCP

[3]As this may be inefficient in terms of packet processing in routers, this lookup may also be performed in edge routers. These may then set TOS bits in the IP header if the mechanism is to be used in routers which may not allow for sequence number lookup.

[4]Misbehaving users should not be able to infer the initial sequence numbers [25].

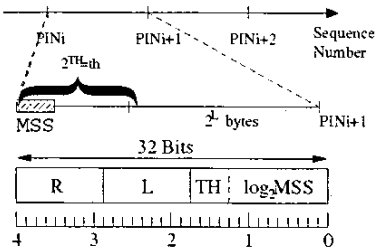

Fig. 3. Structure of the sequence number for RuN2C

connection is established. This scheme permits to infer the priority of the packet by a simple mask based comparison, since when the sequence number belongs to the low priority range $[PIN_i + th, PIN_{i+1}]$ the L intermediate bits will be equal to 0. Packet sequence numbers from a given connection will overflow to the next PIN after 2^{L+TH} bytes which we show in the Subsection III-A can be chosen quite large.

Note that since the sequence number is counted in bytes, the interval 2^{TH} is divided into MSS (Maximum Segment Size) disjoint sets. This allows us to choose also at random the first $log_2 MSS$ bits of the initial sequence number. We can thus increase the randomness of the scheme allowing both the first R bits and the $log_2 MSS$ least significant bits to be random.

As our implementation retains the per byte nature of TCP flow control, the number of high priority bytes of each connection will be fixed and independent of the MSS of the connection. In packet count based schemes, e.g. [13], the number of high priority segments is fixed and the number of high priority bytes depends on the MSS. Our scheme provides positive features from both the user and network point of view. From the network point of view, the maximum number of segments per connection that will get high priority can be controled just by setting the values of TH based on a small data segment (typically $512bytes$). From the user point of view, since the number of bytes with high priority will be fixed by the network, this scheme is transparent and will guarantee a well-defined fairness with respect to other users and at the same time it will prevent misbehaving users from getting preferential treatment since whatever the value of MSS a connection choses, only the number of bytes defines the amount of preferential treatment obtained.

A. Dividing the sequence number space

The threshold th must be chosen in such a way that short flows benefit from the differentiation mechanism while keeping the load of high priority low in order not to harm longer flows. To find a compromise solution, we note two facts. First, short TCP flows are prone to timeouts upon packet losses (see for example [26]). The impact of timeouts can be extremely important on the response time of short flows since its minimum value is 1 s [11]. Thus, to avoid timeouts, packets should be given priority until the congestion window reaches a value of 3 or 4. This is the case if approximately 8 packets are transmitted. This corresponds to a threshold th of 12KBytes if $MSS = 1460$ KBytes. Second, since TCP flow sizes are heavy tailed, even though flows shorter than 8 packets may represent a significant number of TCP flows, they will account for a small proportion of the total load. Hence, giving priority to short flows, will not lead to the starvation of longer flows.

To allow for $th = 12kByte$ we require $TH = log_2(12000) \approx 14$ bits for the least significant bits in the sequence number. Let us consider L to be 8. Sequence numbers will overflow to the next PIN after having sent $2^{22} \approx 4Mbytes$. The latter amount of data corresponds to 2800 packets. At the worst case, this might induce a packet loss with probability 3.6×10^{-4}, which is negligible in respect to the current packet loss rate in the Internet. With this setting there are $2^{10} = 1024$ possible values for PIN. This may not provide satisfactory degree of randomness in the choice of the initial sequence numbers. As mentioned previously, this can be improved choosing the first $log_2 1460 \approx 10$ bits at random. Therefore, the total number of random bits available is equal to $10 + 10 = 20$, which can produce 1 million random values.

B. Deployment

We now give some indications on deploying such a mechanism in a TCP/IP network. Standard TCP connections may be sharing RuN2C enabled routers. In such a case they will not benefit from the priority queue except if they randomly start their sequence number to do so. Analytical models in the previous section and the simulation results given in the next section, show that long TCP connections obtain equivalent performances with tail drop routers and RuN2C routers as long as the load of the priority traffic remains small. One may expect for a similar result to stand for non-adapted short TCP flows if the priority traffic is sufficiently small. RuN2C routers can thus be considered compatible with current TCP implementations. In addition TCP connections implementing RuN2C are not affected by tail drop routers. We thus conclude, that RuN2C can be progressively deployed on a network, as it is always beneficial for all adapted TCP connections and never worse than the current implementation for non-adapted connections.

IV. Simulations

The two packet level scheduling disciplines (RuN and RuN2C) presented in the previous section were studied analytically on the flow level using PS-type models. In this section we use the NS simulator [27] to study metrics which were not analyzed in the Section II, i.e., the extreme values of the response times and the number of ongoing flows in the system. In NS flow sequence numbers start from zero for each flow and the implementation of RuN and RuN2C schemes is straightforward.

The simulation topology used consists of a 10Mbps bottleneck link serviced by 10 access links. The bottleneck buffer has a finite size of 100 packets. We use two basic, but different simulation settings:

(S1) 10Mbps access links with a total network propagation delay of 6 ms.

(S2) 2Mbps access links with a total propagation delay of 60 ms.

(S1) is chosen to be one where the arrival process to the bottleneck link is bursty and a single connection may completely use the bottleneck capacity. Thus the traffic in the network will experience many losses and longer queuing delays. In (S2) the arrival process is less bursty, due to the smaller bandwidth of the access links, and the delay due to propagation delay is more dominant than the delay due to queuing. In all simulations, packet size is set to 1500 bytes and maximum TCP window is 30 packets. We use in RuN2C a threshold of 20 packets.

In the simulations, the schemes are implemented by inspecting the sequence number of the packet from the TCP header at the bottleneck router. The packet is then placed in the buffer either in ascending order according to its sequence number (RuN) or classified to higher or lower priority level (RuN2C) based on a predefined sequence number threshold. In both scheduling mechanisms, the tail drop is in a push out fashion, i.e. if the buffer is full, the arriving packet is first placed into the buffer, and then the tail of the buffer is pushed out. The two proposed scheduling mechanisms are then compared to a traditional FIFO Drop Tail queue.

We consider two flow size distribution scenarios:

(D1) A heavy tailed distribution, which is a mixture of two distributions. 80% of flows are short TCP flows with exponentially distributed flow sizes with mean size of 4 packets (6 kbytes) and 20% are long TCP flows with a bounded Pareto distribution $BP(k, p, \alpha) = BP(13, 5000, 1.1)$ and mean 64 packets. Thus the load of short flows is 20% and the load of long flows 80%.

(D2) 50 persistent TCP flows and a 20% load of short TCP flows with exponentially distributed flow sizes with mean size of 6 packets (9 kbytes).

We use the following metrics to assess the performance of the different mechanisms:

- number of flows in the network,
- mean and maximum latency for short TCP flows,
- throughput and its variance for long TCP flows.

A. Mixed distribution of short and long flows

Consider topology (S1) with traffic distribution (D1). In Figure 4 we plot the number of flows in the system under the three different schedulers as a function of time, thus depicting the change in the stability of the network. RuN and RuN2C mechanisms are able to considerably reduce both the average number of flows in the network as well as the variability. From the figures we thus notice that RuN2C is nearly as efficient as RuN to reduce the average overall number of flows.

Figure 5 shows the change in delay under the three schedulers. We can make two observations from the figure. The mean response time is reduced using the RuN and RuN2C schedulers by more than a ten fold for all flows under the threshold of 20 packets and considerably for flows less than 40 packets i.e. 60kbytes. Thus the proposed scheduling

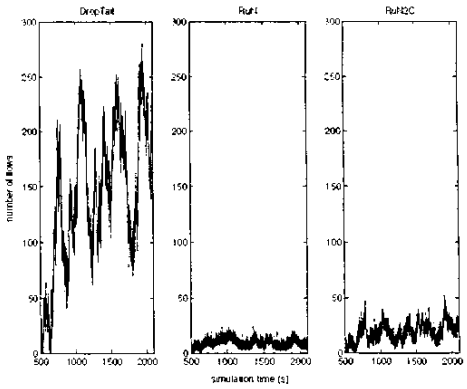

Fig. 4. Number of flows in the network during the simulation under setting (S1) and distribution (D1)

mechanisms are able to reduce the delay for the short flows. Figure 5 also nicely illustrates the difference between the

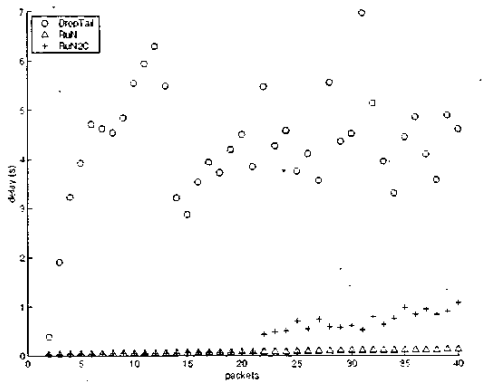

Fig. 5. Mean delay as a function of packet size for short flows under setting (S1) and distribution (D1)

two RuN schemes. Up to the threshold of 20 packets there is minimal difference in strictly ordering the packets and in merely dividing them into two classes. After the threshold the RuN2C scheme resembles the DropTail though for flow sizes close to 20 packets the effect of the preferential treatment to the first 20 packets is still visible in a reduced delay.

Besides the average delay, we study the maximum delay for the short flows shown in Figure 6. We notice that under the RuN and RuN2C schemes the maximum delay is close to the mean delay for all short flows, while the DropTail scheme results in some flows having very large delays. As an example consider the flow with size of 7 packets, under the RuN and RuN2C schemes it has a maximum delay of 100 ms, while under DropTail the maximum delay is 200s, a difference of 3 orders. This is the result of multiple backoffs of the TCP retransmission timer. This example illustrates the blackout impression users can have even in a moderately loaded system. Thus, RuN and RuN2C schemes significantly reduce the timeouts for short flows and hence improve the predictability of response times.

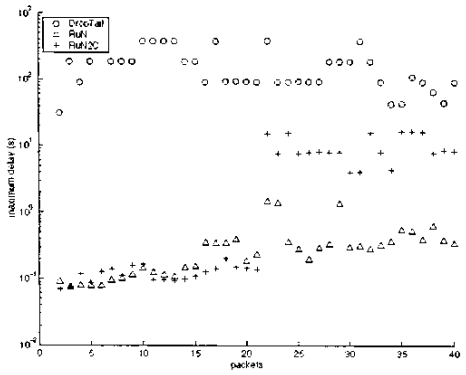

Fig. 6. Maximum delay as a function of packet size for short flows under setting (S1) and distribution (D1)

Figure 7 depicts how the gain in smaller delay for the short flows affects the throughput of long flows. It seems that for a maximum flow size of 5000 packets i.e. 7.5Mbytes, all flows have the same or better throughput under the RuN and RuN2C schemes compared to DropTail. However, as will be shown later the RuN scheme may not be beneficial for all long flows if many long flows coincide at the same time.

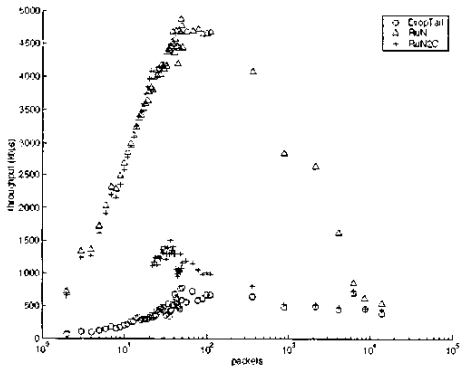

Fig. 7. Mean throughput as a function of packet size for all flows under setting (S1) and distribution (D1)

Consider now topology (S2), with smaller access link capacities and a larger overall RTT. It is expected that the gain in smaller delay for the short flows is not as considerable as there are fewer drops and thus fewer retransmits in topology (S2), furthermore the propagation delay dominates the total delay for flows. Figure 8 depicts the number of flows in the network. Though the aggregated packet arrival process to the bottleneck link is less bursty, there is still a difference in the stability of the network.

Figure 9 shows the mean response time using the two schedulers. Though the RuN and RuN2C schemes are able to reduce the delay, the effect of slow start (step like pattern) and the large propagation delay are dominant.

We notice that the change in average delay is still considerable. Mainly we notice that the delay under DropTail is of the same magnitude for a propagation delay of 6 ms

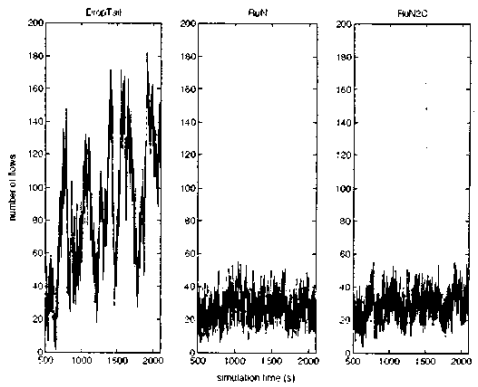

Fig. 8. Number of flows in the network during the simulation under setting (S2) and distribution (D1)

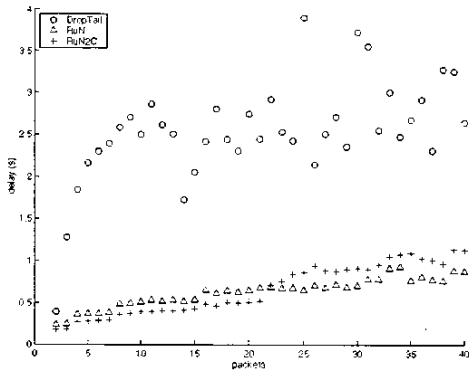

Fig. 9. Mean delay as a function of packet size for short flows under setting (S2) and distribution (D1)

or 60 ms. While for RuN and RuN2C schemes the delay is clearly mainly due to propagation delay and marginally due to queuing delays. The maximum delay is also reduced for the short flows, as shown in Figure 10.

Figure 11 depicts that as the change in delay for the short flows was not as large, neither is the change in throughput for the long flows. However, it still seems that the RuN2C scheme is able to improve the throughput for all flows.

B. Effect on persistent flows

Now consider the same two network parameter settings as previously, but the flow distribution (D2). Instead of long TCP flows that arrive randomly with a fixed size we load the network with 50 persistent TCP flows. We then study how the scheduling mechanisms affect these persistent flows and the randomly arriving short flows that coexist with the persistent flows. We present the same set of metrics as in the previous section. The effect of the RuN schemes on the persistent flows is different than on the randomly arriving long flows. However, studying the persistent flows gives us an indication of the effect of scheduling during a snapshot of the network, when long flows coexist at the same time.

Figure 12 depicts the number of flows in the network on the log scale, as the reduction in the number of short flows is

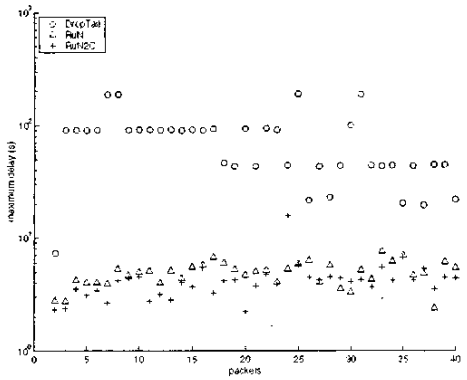

Fig. 10. Maximum delay as a function of packet size for short flows under setting (S2) and distribution (D1)

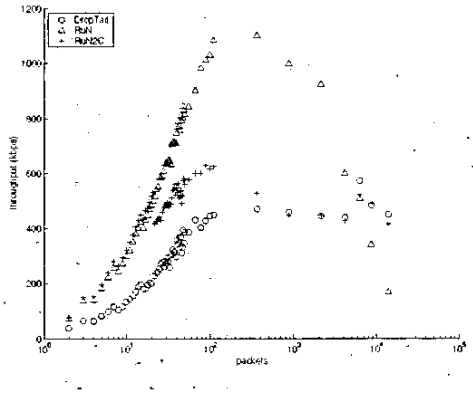

Fig. 11. Mean throughput as a function of packet size for all flows under setting (S2) and distribution (D1)

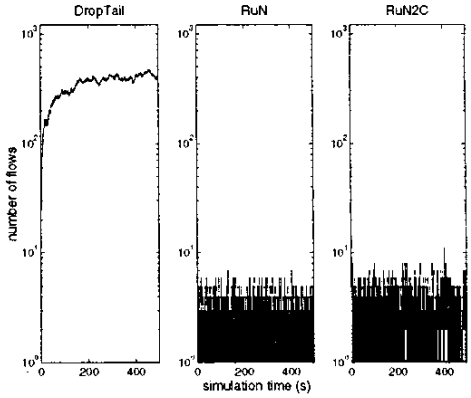

Fig. 12. Number of short flows in the network during the simulation under topology (S1) and distribution (D2)

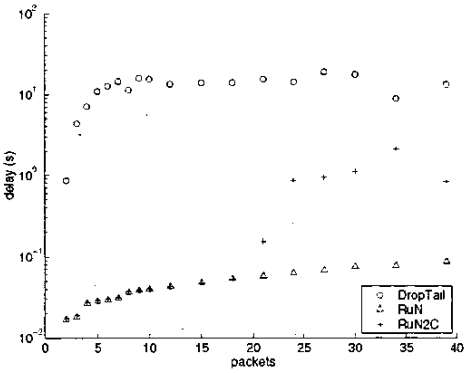

Fig. 13. Mean delay as a function of packet size for short flows under topology (S1) and distribution (D2)

of more than two orders.

Figure 13 shows the change in average delay for short flows under the three schedulers. The stabilizing effect of the RuN schemes is clear from figures 12 and 13. Under DropTail the latency for short flows is dominated by drops and retransmits as it has been already mentioned in Section IV-A.

Figure 14 depicts how the gain in smaller delay for the short flows affects the throughput of persistent flows. From the figure we see that the 50 persistent flows have negligible difference between the mean throughputs, though under RuN the throughput is the lowest. There are, however, large differences in the variance of the throughput. The RuN scheme reduces the mean throughput and has zero variance. This is because the RuN scheme services the persistent flows exactly the same amount and as a consequence the data points in the middle figure all coincide and look like one data point! Under RuN, all the persistent flows start at the same time and are serviced in the order of packet sequence numbers in a strict round robin fashion packet by packet. If at some point a more aggressive flow has more packets serviced than others, it has to wait in the buffer or is pushed out of the buffer and thus slowed down until the others have caught up with the aggressive flow. This shows how the LAS policy may reduce the overall mean response time if all long flows start at the same time, but may be extremely unfair for long flows starting during the service time of other long flows, as discussed in Section II.

Note also that the throughput variance for RuN2C is three times smaller than the throughput variance for DropTail. The RuN2C scheme is thus able to both increase the average throughput as well as reduce the variance, without the risk of the pathological behavior of LAS. For the topology (S2) the conclusions are the same and figures are omitted due to lack of space.

V. Conclusions

We propose a packet level size-based scheduling mechanism for TCP flows, RuN2C, which is a threshold based differentiation scheme. We describe an implementation of this mechanism which has the advantage of being TCP compatible, robust, scalable and progressively deployable. We show numerically, how the value of the threshold affects the optimality of the scheduling policy and compare by means of simulations the proposed implementation to standard TCP over tail drop buffers and to LAS implementation RuN.

We showed that RuN2C is beneficial from both the system's and the user's point of view for small transfers. From the user's perspective the network response becomes more predictable

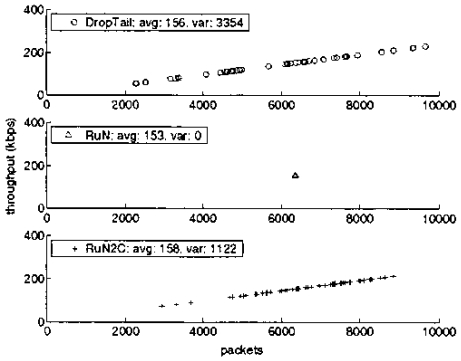

Fig. 14. Mean throughput as a function of packet size for persistent flows under topology (S1) and distribution (D2)

and the average response time is reduced. In addition, many short flows get served promptly at the expense of some few long flows. In fact, we show that RuN2C performs very similar to standard TCP for large flows. From the system's point of view, the average number of active sessions is reduced.

On the contrary, we show that LAS based implementations, on top of being difficult to implement, may show degraded performance for large transfer sizes since for service distributions with infinite variance LAS deviates increasingly from PS for large file sizes. This is in contrast to RuN2C, which exhibits performance close to that of standard TCP for large transfers and is simpler to implement. At the flow level, we prove that the $PS + PS$ scheduling policy reduces the mean overall response time $E[T^{PS+PS}]$ with respect to PS when the hazard rate of the service time distribution is decreasing. In addition, we provide numerical indications that by an appropriate tuning of the threshold, $PS+PS$ performs very closely to LAS, which is known to be the optimal scheduling discipline, with respect to the average time in the system, among work conserving disciplines, when no precise knowledge of the job lengths is available and when the hazard rate of the job distribution is decreasing.

At the TCP level the benefit brought by the RuN2C mechanism seems to come at no cost. This benefit is obtained as a consequence of a better work scheduling, from the performance point of view, of the loss epochs. That is, short TCP flows are extremely vulnerable to packet loss and they have a high probability of experiencing a costly timeout. In contrast, large TCP flows are more robust against losses since they are more likely to have large congestion windows. Therefore, many short sessions benefit from giving priority to short flows at almost no cost for long flows, as short sessions account for a small proportion of total network load.

ACKNOWLEDGMENTS

The visit of Eeva Nyberg to INRIA Sophia-Antipolis was financed by INRIA ARC-TCP grant. Eeva Nyberg is also partly supported by the Academy of Finland, and the Nokia and TES foundations.

REFERENCES

[1] U. Ayesta and K.E. Avrachenkov, "The effect of the initial window size and limited transmit algorithm on the transient behavior of TCP transfers," in *15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, Wurzburg, Germany, July 2002.

[2] N. Bhatti, A. Bouch, and A.J. Kuchinsky, "Integrating user-perceived quality into web server design," in *Proceedings of WWW'00*, Amsterdam, The Netherlands, 2000.

[3] Leonard Kleinrock, *Queueing Systems, vol. 2*, John Wiley and Sons, 1976.

[4] S.F. Yashkov, "On feedback sharing a processor among jobs with minimal serviced length (in russian)," *Technika Sredstv Svyazi, Ser. ASU*, vol. 2, pp. 51–62, 1978.

[5] S.F. Yashkov, "Processor-sharing queues: Some progress in analysis," *Queueing Systems*, vol. 2, pp. 1–17, 1987.

[6] R. Righter and J.G. Shanthikumar, "Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures," *Probability in the Engineering and Informational Sciences*, vol. 3, pp. 323–333, 1989.

[7] A. Wierman, N. Bansal, and M. Harchol-Balter, "A note on comparing response times in the M/G/1/FB and M/G/1/PS queues," *Operations Research Letters*, vol. 32, no. 1, pp. 73–76, 2004.

[8] S. Ben Fredj, T.Bonald, A.Proutiere, G.Regnie, and J.Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," in *Proceedings of ACM SIGCOMM*, Pitsburgh, PA, 2002.

[9] Kevin Thompson, Gregory J. Miller, and Rick Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, 1997.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurement from the Sprint IP backbone," *IEEE Network Magazine*, no. to appear, 2003.

[11] Vern Paxson and Mark Allman, "Computing TCP's retransmission timer," RFC2988, Nov. 2000.

[12] Wael Noureddine and Fouad Tobagi, "Improving the performance of interactive TCP applications using service differentiation," in *Proceedings of IEEE INFOCOM*, New York, NY, 2002.

[13] Lian Guo and Ibrahim Matta, "The war between mice and elephants," in *Proceedings of the 9th IEEE International Conference on Network Protocols ICNP'01*, 2001.

[14] Liang Guo and Ibrahim Matta, "Differentiated control of web traffic: A numerical analysis," in *Proceedings of SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, Boston, MA, 2002.

[15] Liang Guo and Ibrahim Matta, "Scheduling flows with unknown sizes: Approximate analysis," Tech. Rep. BU-CS-2002-009, Boston University, Mar. 2002.

[16] I. Rai, G. Urvoy-Keller, and E. Biersack, "Size-based scheduling with differentiated services to improve response time of highly varying flows," in *Proceedings of the 15th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management*, Wurzburg, Germany, 2002.

[17] I. Rai, G. Urvoy-Keller, and E. Biersack, "Analysis of LAS scheduling for job size distributions with high variance," in *Proceedings of ACM Sigmetrics 2003, International Conference on Measurement and Modeling of Computer Systems*, 2003.

[18] L. Massoulié and J. Roberts, "Bandwidth sharing and admission control for elastic traffic," *Telecommunication Systems*, no. 15, pp. 185–201, 2000.

[19] E.G. Coffman, "Stochastic models of multiple and time-shared computer operation," Tech. Rep. 66-38, Department of Engineering, University of California at Los Angeles, June 1966.

[20] E.G. Coffman and L. Kleinrock, "Feedback queueing models for time-shared systems," *Journal of the Association for Computing Machinery*, no. 15, pp. 549–576, 1968.

[21] L.E. Schrage, "The queue M/G/1 with feedback to lower priority queues," *Management Science*, no. 13, pp. 466–471, 1967.

[22] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic convergence of scheduling policies with respect to slowdown," *Performance Evaluation*, no. 49, pp. 241–256, 2002.

[23] K. Avrachenkov, U. Ayesta, and P. Brown, "Batch arrival M/G/1 processor sharing with application to size-based scheduling," Tech. Rep., INRIA Technical Report RR-5043, 2003.

[24] S. Aalto, U. Ayesta, and E. Nyberg, "Multilevel processor-sharing scheduling disciplines: mean delay analysis," Tech. Rep., Helsinki University of Technology, 2003.

[25] S. Bellovin, "Defending against sequence number attacks," RFC 1948, May 1996.
[26] Hari Balakrishnan, Venkata Padmanabhan, Srinivasan Seshan, Mark Stemm, and Randy Katz, "TCP behavior of a busy web server: analysis and improvements," in *Proceedings of IEEE INFOCOM*, San Francisco, CA, 1998.
[27] "Network simulator, ver.2, (ns-2)," available at http://www-mash.cs.berkeley.edu/ns.

APPENDIX A: PROOF OF THEOREM 1

First we note that the solution of the integral equation (8) is equivalent to the solution

$$\begin{aligned}\delta\alpha'(x) &= \lambda\overline{n}\int_0^\infty \delta\alpha'(y)B(x+y)dy \\ &\quad +\lambda\overline{n}\int_0^x \delta\alpha'(y)B(x-y)dy \\ &\quad +bB(x),\end{aligned} \tag{9}$$

where

$$\delta\alpha'(x) := \alpha'(x) - \frac{1-\rho_{th}}{1-\rho}. \tag{10}$$

The above transformation removes the constant component of the solution. Next, we consider the fixed point iterations

$$\begin{aligned}\delta\alpha'_{k+1}(x) &= \lambda\overline{n}\int_0^\infty \delta\alpha'_k(y)B(x+y)dy \\ &\quad +\lambda\overline{n}\int_0^x \delta\alpha'_k(y)B(x-y)dy \\ &\quad +bB(x), \quad k=0,1,...\end{aligned}$$

on the complete functional space of continuous bounded non negative functions $\mathcal{C}[0,\infty)$ with the supremum metric. Let $\|\delta\alpha'\| = \sup_x |\delta\alpha'(x)| < \infty$.

If we show that the linear integral operator

$$\begin{aligned}\mathcal{A}[\beta(x)] &= \lambda\overline{n}\int_0^\infty \beta(y)B(x+y)dy \\ &\quad +\lambda\overline{n}\int_0^x \beta(y)B(x-y)dy + bB(x)\end{aligned}$$

is a contraction, then the integral equation (8) has a unique solution in $\mathcal{C}[0,\infty)$. Clearly the operator $\mathcal{A}[\beta(x)]$ maps the space $\mathcal{C}[0,\infty)$ into itself.

Let us denote as d the distance in the metric space $\mathcal{C}[0,\infty)$, that is $d(\delta\alpha'_1,\delta\alpha'_2) = \sup_x\{|\delta\alpha'_1 - \delta\alpha'_2|\}$. Let us now show that the linear operator $\mathcal{A}[\beta(x)]$ is indeed a contraction mapping on $\mathcal{C}[0,\infty)$.

$$\begin{aligned}d(\mathcal{A}[\delta\alpha'_1],\mathcal{A}[\delta\alpha'_2]) &= \sup_x\{|\mathcal{A}[\delta\alpha'_1] - \mathcal{A}[\delta\alpha'_2]|\} \\ &\le \lambda\bar{n}\sup_x\{\mathcal{A}[|\delta\alpha'_1] - \mathcal{A}[\delta\alpha'_2]|\} \\ &\quad \sup_x\left(\int_0^\infty B(x+y)dy + \right. \\ &\quad \left.\int_0^x B(x-y)dy\right) \\ &= \lambda\bar{n}d(\delta\alpha'_1,\delta\alpha'_2)\frac{\overline{X}-\overline{X}_{th}}{1-\rho_{th}} \\ &= d(\delta\alpha'_1,\delta\alpha'_2)\frac{\rho-\rho_{th}}{1-\rho_{th}}.\end{aligned}$$

Thus, the mapping is a contraction if $\rho < 1$.

We now show that $\lim_{x\to\infty}\delta\alpha'(x) = 0$. We note that $\lim_{x\to\infty} B(x) = 0$. Clearly the first integral tends to zero as $x\to\infty$. Then we have:

$$\limsup_{x\to\infty}\delta\alpha'(x) = \frac{\lambda\overline{n}}{1-F(th)}$$

$$\limsup_{x\to\infty}\int_0^x \delta\alpha'(y)(1-F(x-y+th))dy$$

We note that $1-F(x-y+th)$ is an increasing function on y and $\lim_{y\to\infty}(1-F(x-y+th)) = 1$. We choose $x^* < x$ to be large enough so $\forall x > x^*$, $\delta\alpha'(x) < \limsup_{z\to\infty}\delta\alpha'(z)$. Let us denote $1-F(x-x^*+th) = \varepsilon$, then

$$\begin{aligned}\delta\alpha'(x) &\le \frac{\lambda\overline{n}}{1-F(th)}\left(x^*\varepsilon M + \limsup_{z\to\infty}\delta\alpha'(z)\right) \\ &\quad \int_{x^*}^x (1-F(x-y+th))dy.\end{aligned}$$

Now we take the $\limsup_{x\to\infty}$ of the above expression. We note that the first component tends to 0 since $\limsup_{x\to\infty}\varepsilon = \lim_{x\to\infty}(1-F(x-x^*+th)) = 0$. Integrating by parts it is easy to see that $\int_0^x(1-F(y))dy = \bar{X}_x$. Then we obtain

$$\begin{aligned}\limsup_{x\to\infty}\delta\alpha'(x) &\le \frac{\lambda\overline{n}}{1-F(th)}\limsup_{z\to\infty}\delta\alpha'(z) \\ &\quad \limsup_{x\to\infty}\int_{x^*}^x (1-F(x-y+th))dy \\ &= \frac{\lambda\overline{n}}{1-F(th)}\limsup_{z\to\infty}\delta\alpha'(z) \\ &\quad \limsup_{x\to\infty}\frac{\bar{X}_{x-x^*-th} - \bar{X}_t h}{1-F(th)} \\ &= \frac{\rho-\rho_{th}}{1-\rho_{th}}\limsup_{x\to\infty}\delta\alpha'(x).\end{aligned}$$

Since $(\rho-\rho_{th})/(1-\rho_{th}) < 1$ if $\rho < 1$, the equality holds only if $\limsup_{x\to\infty}\delta\alpha'(x) = 0$. Consequently, the limit exists and is equal to zero. Hence, using equations (10) and (7), we conclude that $\lim_{x\to\infty}\overline{S}^{PS+PS}(x) = 1/(1-\rho)$.

APPENDIX B: PROOF OF THEOREM 2

Let $\mathcal{S}$ be the set of scheduling disciplines that do not use any information about the remaining service times of jobs, but only information on the attained services of these jobs. Note that $PS+PS, PS, LAS \in \mathcal{S}$. Define $\overline{U}_x^S$ as the mean unfinished truncated work in a system with a scheduling discipline $S\in\mathcal{S}$. For example, let us consider a job present in the system which has obtained already 3 units of service and that requires 10 more units of service. If $x = 5$, the contribution of this job to the unfinished truncated work will be equal to 2. It has been shown recently [24] that for any scheduling $S\in\mathcal{S}$ $\overline{U}_x^S$ is given by

$$\overline{U}_x^S = \lambda\int_0^x \overline{T}^S(y)(1-F(y))dy. \tag{11}$$

Furthermore, in [24] (see Proposition 1) it has been shown that for any $S_1, S_2 \in \mathcal{S}$, where $\overline{U}_x^{S_1} \le \overline{U}_x^{S_2}$ and the

service time distribution has a decreasing hazard rate, then $E[T^{S_1}] \leq E[T^{S_2}]$. In the rest of the proof we show that indeed $\overline{U}_x^{PS+PS} \leq \overline{U}_x^{PS}$.

Recalling the results of equations (6) and (7) we have that

$$\overline{T}^{\mathrm{PS+PS}}(x) = \begin{cases} \dfrac{x}{1-\rho_{th}} & \text{if } x \leq th, \\ \dfrac{\overline{W}(th)+th}{1-\rho_{th}} + \dfrac{\alpha(x-th)}{1-\rho_{th}} & \text{if } x > th. \end{cases} \tag{12}$$

Denote now

$$\alpha^* = \inf_{x>0} \alpha'(x).$$

The following inequality follows straightly from the integral equation (8):

$$\begin{aligned} \alpha^* &= \inf_{x>0} \{\lambda\overline{n} \int_0^\infty \alpha'(y)B(x+y)\,dy \\ &\quad + \lambda\overline{n} \int_0^x \alpha'(y)B(x-y)\,dy + + bB(x) + 1\} \\ &\geq \inf_{x>0} \{\lambda\overline{n} \int_0^\infty \alpha^* B(x+y)\,dy \\ &\quad + \lambda\overline{n} \int_0^x \alpha^* B(x-y)\,dy + 1\} \\ &= \lambda\overline{n}\alpha^* \int_0^\infty B(z)\,dz + 1 \\ &= \alpha^* \frac{\rho - \rho_{th}}{1-\rho_{th}} + 1. \end{aligned}$$

and thus,

$$\alpha^* \geq \frac{1-\rho_{th}}{1-\rho}. \tag{13}$$

By combining equations (12) and (13), we have

$$\begin{cases} (\overline{T}^{\mathrm{PS+PS}})'(x) = \dfrac{1}{1-\rho_{th}} \leq \dfrac{1}{1-\rho} & \text{if } x < th, \\ (\overline{T}^{\mathrm{PS+PS}})'(x) \geq \dfrac{\alpha^*}{1-\rho_{th}} \geq \dfrac{1}{1-\rho} & \text{if } x > th. \end{cases} \tag{14}$$

Note further that

$$(\overline{T}^{\mathrm{PS}})'(x) = \frac{1}{1-\rho}. \tag{15}$$

Since $\overline{T}^{\mathrm{PS}}(0) = \overline{T}^{\mathrm{PS+PS}(a)}(0) = 0$, it follows from (14) and (15) that, for all $x < th$,

$$\overline{T}^{\mathrm{PS+PS}}(x) \leq \overline{T}^{\mathrm{PS}}(x). \tag{16}$$

Define then

$$x^* = \inf\{x \geq th \mid \overline{T}^{\mathrm{PS+PS}}(x) > \overline{T}^{\mathrm{PS}}(x)\}.$$

By the above definition and (16), we have, for all $x < x^*$,

$$\overline{T}^{\mathrm{PS+PS}}(x) \leq \overline{T}^{\mathrm{PS}}(x).$$

Thus, by (11), we deduce that, for all $x \leq x^*$,

$$\overline{U}_x^{PS+PS} \leq \overline{U}_x^{PS}.$$

By definiton of x^* and using equations (14) and (15), we have that, for all $x > x^*$,

$$(\overline{U}_x^{\mathrm{PS+PS}})' = \lambda\overline{F}(x)\overline{T}^{\mathrm{PS+PS}}(x) > \lambda\overline{F}(x)\overline{T}^{\mathrm{PS}}(x) = (\overline{U}_x^{\mathrm{PS}})'. \tag{17}$$

Finally, since both $PS+PS$ and PS are work conserving disciplines, for which the mean unfinished work is equal in the M/G/1 queue, we have

$$\lim_{x\to\infty} \overline{U}_x^{PS+PS} = \lim_{x\to\infty} \overline{U}_x^{PS}. \tag{18}$$

Now we argue that, for all $x > x^*$,

$$\overline{U}_x^{PS+PS} < \overline{U}_x^{PS}.$$

Suppose on contrary that there exists x^{**} such that $\overline{U}_{x^{**}}^{PS+PS} = \overline{U}_{x^{**}}^{PS}$. Then, consider the following integral that is strictly positive by equation (17)

$$\begin{aligned} 0 &< \int_{x^{**}}^\infty \left((\overline{U}_x^{PS+PS})' - (\overline{U}_x^{PS})'\right) dx \\ &= \lim_{x\to\infty} (\overline{U}_x^{PS+PS} - \overline{U}_x^{PS}) = 0. \end{aligned}$$

The last equality follows from equation (18). Hence by contradiction, for all $x > x^*$, $\overline{U}_x^{PS+PS} < \overline{U}_x^{PS}$, which completes the proof.