Institut Eurécom
Department of Corporate Communications
2229, Route des Cretes
B.P. 193
06904 Sophia-Antipolis FRANCE

Research Report RR-03-075

Analyzing the Performance of TCP Flows in Packet Networks with LAS Schedulers

September 2003

Idris A. Rai, Ernst W. Biersack, Guillaume Urvoy-Keller

Tel : (+33) 4 93 00 26 26
Fax : (+33) 4 93 00 26 27
Email : rai@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members : Bouygues Télécom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

Abstract

The networking community has focused mainly on Processor Sharing (PS) policies in routers in order to establish QoS guarantees. As of today, PS scheduling has not been deployed in routers and the Internet still uses FIFO scheduling that offers only a best-effort service. Also, Internet traffic exhibits a high variability in terms of the flow sizes: Most of the flows are short, while more than half of the bytes are carried by a few largest flows. Given these observations, we recommend Least Attained Service (LAS) scheduling in routers, which favors short flows by giving the highest priority to the first packets of each flow. As a result, LAS improves the overall user perceived performance. Using simulation we show that under LAS scheduling, as compared to FIFO, the transmission time and loss rate for short TCP flows are significantly reduced, with a negligible increase in transmission time for the largest flows. We also show that the improvement seen by short TCP flows under LAS is mainly due to the way LAS interacts with TCP algorithm in the slow start (SS) phase, which result in shorter round-trip times and very low packet loss rates for short flows. We also investigate the performance of long-lived TCP and UDP flows under LAS and compare the results to FIFO scheduling.

1 Introduction

Scheduling algorithms have been studied for decades and many scheduling policies were first developed in the context of job scheduling in operating systems. The books by Conway [12], Coffman and Denning [11], and Kleinrock [26] describe and analyze a host of scheduling polices. The most common performance metric that is used to analyze the performance of scheduling algorithms is job **response time**, which comprises both, the waiting and the processing time, and is defined as the difference between the completion time and the arrival time of a job. In the last two decades, *packet* scheduling in routers has been an active area of research and most of the attention has been focused on Processor Sharing type of scheduling algorithms [30, 15, 42]. Processor Sharing has many interesting properties: It assures a max-min fair allocation of bandwidth among competing flows[1], provides protection against misbehaving flows [25], and allows to establish end-to-end delay bounds for individual flows [33].

Despite all these properties, PS scheduling has not been deployed. Instead, routers still implement FIFO scheduling and possibly active queue management such as RED. As a consequence, the Internet only provides a *best-effort* service. TCP remains the most widely used transport protocol, not only for data transfer, but also for media streaming [39]. Most of the TCP flows are short, while more than 50% of the bytes are carried by less than 5% of the largest flows [35, 13]. Given these facts, we propose to study size-based scheduling policies in conjunction with their interaction with TCP flows in order to improve the mean transmission (response) time of TCP flows.

We know from timesharing systems that there exist scheduling policies that offer low response times to interactive jobs without starving large batch jobs. One such policy is *Least Attained Service* (LAS), which is also referred to as *Foreground-Background* (FB) [26] or *Shortest Elapsed Time* (SET) first [11] scheduling. LAS is a preemptive scheduling policy that favors short jobs without prior knowledge of job sizes. To this end, LAS gives service to the job in the system that has received the least service. In the event of ties, the set of jobs having received the least service shares the processor in a processor-sharing mode. A newly arriving job always preempts the job currently in service and retains the processor until it departs, or until the next arrival appears, or until it has obtained an amount of service equal to that received by the job preempted on arrival, whichever occurs first. In the past, LAS has been believed to heavily penalize large jobs. However, it has recently been shown that the mean response time of LAS

[1] A flow is defined as a group of packets with a common set of attributes.

highly depends on the job size distribution [36]. In particular, for job size distributions with a large coefficient of variation [2], the improvement in mean response time seen by the short jobs comes at the expense of a negligible increase in mean response time for the largest jobs.

These results make us consider LAS for packet scheduling in routers. However, the term job is commonly used in scheduling theory to denote a piece of workload that arrives to the system all *at once*. Given this definition, a flow in a packet network cannot be considered as a job since a flow does not arrive at the system at once. Instead, the source transmits a flow as a sequence of packets, possibly spaced out in time, that are in turn statistically multiplexed with packets from other flows. Also, when TCP is used as transport protocol, due to the closed-loop nature of TCP, the treatment given to an individual packet may affect the temporal behavior of the remaining packets in the flow. For these reasons, the analytical results on LAS derived for jobs cannot be directly be applied to flows, and we use simulation to investigate the performance of LAS for flows.

In the context of flows, the next packet serviced by LAS is the one that belongs to the flow that has received the least service. By this definition, LAS gives all the available bandwidth to a newly arriving flow until it receives the service equal to the least service received by a flow in the system before its arrival. If two or more flows have received an equal amount of service, they share the system resources fairly.

Empirical studies have shown that Internet traffic consists of many short flows with 10-20 packets on average [10] and a few very large flows that contain a large fraction of the packets. These short flows are mainly Web data transfers triggered by user interaction. As the use of WWW remains popular, Web traffic will continue to make up a significant fraction of the Internet traffic. Web traffic is transferred using the TCP transport protocol. However, a look at the behavior of TCP in a network with FIFO routers reveals the factors that affect the performance of *short* flows: i) Regardless of the available bandwidth in the network, TCP conservatively initializes its sending window to one packet and then doubles the window for every window worth of ACKs received until congestion is detected (this behavior is referred to as slow start (SS)). Thus, even if the network has sufficient bandwidth, the duration of a transfer depends very much on *round trip time* (RTT) of the connection. (ii) Since short flows often do not have enough packets in transit to trigger the triple duplicate ACK mechanism *fast retransmission*, they must rely on the *retransmission timeout* (RTO) to detect loss. Packet retransmissions that are triggered by a RTO can significantly prolong the transmission time of a short TCP flow. (iii) TCP uses its own packets to sample the round trip times and to estimate the appropriate RTO value. For the first control packets and the first data packet, the initial RTO value is usually set to 3 seconds. Thus, losing any of these packets can significantly prolong the transmission time of short flows.

To investigate the impact of LAS on the performance of data flows of different sizes (short and large) in network routers we use the network simulator ns-2 [23]. The implementation of LAS in ns-2 involves maintaining a single priority queue and inserting each incoming packet at its appropriate position in the queue. The less service a flow has received so far, the closer to the head of the queue its arriving packets will be inserted. When a packet arrives to a full queue, LAS first inserts the arriving packet at its appropriate position in the queue and then drops the packet that is at the end of the queue.

Our simulation results show that LAS scheduler significantly reduces the mean transmission time and loss rate of short TCP flows as compared to FIFO scheduler, with a small increase in mean response time of large flows. The results also show that a long-lived FTP flow under LAS is not locked out when competing with background Web traffic. While the results show that a long-lived UDP flow, when competing with background Web traffic, experiences higher jitter under LAS than under FIFO, we

[2] Coefficient of variation is defined as the ratio between standard deviation and mean of a distribution.

demonstrate that a small receiver playout buffer is sufficient to ensure that no packet misses the deadline.

This paper is organized as follows; after presenting the related work in the next section, we present preliminary concepts that show how LAS interacts with TCPs congestion control and retransmission mechanisms in Section 3. We then discuss the simulation setting and results for short flows in Section 4. In Section 5 we investigate the performance of long-lived TCP and long-lived UDP flows that compete with Web traffic. We discuss implementation issues of LAS in Section 6 and finally summarize the results and discusses open issues in Section 7.

2 Related Work

A large number of papers propose to improve the performance of short flows by changing TCP. For example [3, 31, 43, 2] suggest to use a larger initial window value to reduce the time short flows spend in the SS phase. Other papers [1, 24] propose rate based pacing of packets to reduce the burstiness of TCP transmissions. Reducing the burstiness reduces the likelihood of multiple packet losses in the same congestion window, which generally degrades the performance of a TCP flow. Other proposals to improve the startup and recovery behavior of TCP include [4, 5, 8, 16, 22]. The work in this paper does not propose to modify TCP. Instead, we propose to use LAS scheduling to resolve the problems short flows experience in the Internet today.

Hence, closely related to our work are network-based approaches that are proposed to improve the performance of short flows. In [18, 9, 28, 41] Diffserv-like architectures are proposed in routers to isolate short flows from large flows. In these approaches, edge routers classify and mark incoming packets as belonging to a short or to a large flow (using a pre-defined threshold). The core routers use the packet marking to separate different flows in order to give preferential treatment to short flows using either selective packet discard mechanisms or priority queueing schemes. Simulation results show that the proposed mechanisms reduce the mean transmission time of short flows by about 30% [9] and by about 80% for medium sized flows of sizes between 50-200 packets [19] compared to simple FIFO with RED. In [18], a modified active queue management is proposed that reduces the packet drop rate of short flows and helps to reduce the transmission time of short and medium flows by 25-30% compared to RED.

One difficulty with threshold-based schemes, which is not addressed in the previous work, is how to tune the threshold value to capture all short flows. In addition, flows that are marked to belong to the same class (short or large flows) are serviced indiscriminately in *FIFO* order. This is maybe the reason why the transmission time of short flows under these approaches is only slightly improved. LAS, in contrast, offers service priority at a much finer level of granularity. For example, the first packets of a flow receive the service under LAS (almost) immediately upon their arrival, whereas under FIFO, they must wait for all packets that are ahead of them in the queue to complete service before being served. Hence, LAS significantly reduces the mean transmission time and loss rate for short flows without the need to set a threshold or the need to use active queue management schemes.

Williamson and Wu [40] recently proposed a Context Aware Transport/Network Internet Protocol (CATNIP) for Web documents. This work is motivated mainly by the fact that loss of the first packets of a TCP flow has a much more negative impact on the transmission time of a flow than loss of other packets. CATNIP uses application layer information, namely the Web document size, to provide explicit context information to the TCP and IP protocols; Using CATNIP, IP routers identify and mark packets of a flow differently to avoid that the most important packets get dropped. Experiments with CATNIP in routers show that it can reduce the mean Web document retrieval time by 20-80%. In contrast to CATNIP, LAS does not require the knowledge of flow sizes and our simulation results indicate that LAS scheduling

significantly outperforms CATNIP.

Finally, the *Shortest Remaining Processing Time* (SRPT) scheduling policy is known to be the optimal policy in the sense that it *minimizes* the mean transmission time of jobs [37]. An experiment where SRPT was implemented in the network interface of a Web server [20] showed that SRPT substantially reduces the mean transmission times of short Web objects. However, unlike LAS, SRPT must know the flow sizes to decide what flow to schedule next.

3 Background

In this section, we highlight the interactions of LAS with a TCP flow that is in SS to explain how LAS scheduler improves the performance of short TCP flows. We will assume Reno version of TCP. It is worth mentioning that we compare LAS scheduling to FIFO scheduling only, because FIFO is the policy that is most widely deployed in networks.

Short flows, e.g., most of Web transfers may be completely transferred during the SS phase. Thus, the performance of these flows is dominated by startup effects such as connection establishment and the conservative nature of the SS phase. However, the TCP SS phase is quite a slow process for short flows because the RTTs during this phase contribute significantly to the transmission times of short flows. In case of packet losses, short flows are most likely to timeout and make use of RTO to recover the losses because short flows generally do not have enough packets to use fast retransmit mechanism to recover lost packets, which we will assume in this section. Using RTO to recover lost packets further deteriorates the transmission time of the flows. Hence, there is a need to pay special attention to short flows from the time they send their very first packets so as to avoid these negative effects. We elaborate on the ways LAS scheduler solves these problems for short flows in the next section.

3.1 The impact of LAS to TCP mechanism of short flows

There are a number of reasons that make LAS a better candidate than FIFO when comes to improving the performance of short TCP flows. By giving highest service priority to the first packets in a flow, LAS prevents the first packets from waiting in the queue, and they see little or no queueing delay, which significantly reduces the RTT of the first packets of each flow. Since the RTT dominates the transmission time of short flows, reducing the RTT means reducing the transmission time of short flows. Figure 1 shows the sequence number of received packets versus time plot that elaborates this behavior under both LAS and FIFO assuming that there is no congestion in the network. The figure shows that a short flow of length L packets transmits its packets faster under LAS than under FIFO, due to lower RTT under LAS than under FIFO. As a result, the transmission time of the same flow under LAS ($T^{LAS}_{no\ loss}$) is lower than the transmission time under FIFO ($T^{FIFO}_{no\ loss}$).

The packet discard mechanism under LAS is such that when the buffer is full, LAS first inserts an arriving packet in the queue and then drops the packet that is at the tail of the queue, i.e., LAS gives buffer space priority to short flows. This significantly reduces packet losses during the initial SS phase for all flows, regardless of their sizes. Instead, LAS mainly drops packets from large flows that have already transmitted a certain amount of data (have low service priority). Hence under LAS there is hardly a need for packet error recovery to rely on the RTO to detect packet loss. This is an important feature of LAS for short flows. Figure 2 shows the sequence number versus time plot for a short flow of size L packets that does not experience losses under LAS but losses the $(L_l + 1)$th packet under FIFO. The short flow under FIFO retransmits the packet after time RTO^{FIFO}, which significantly prolongs

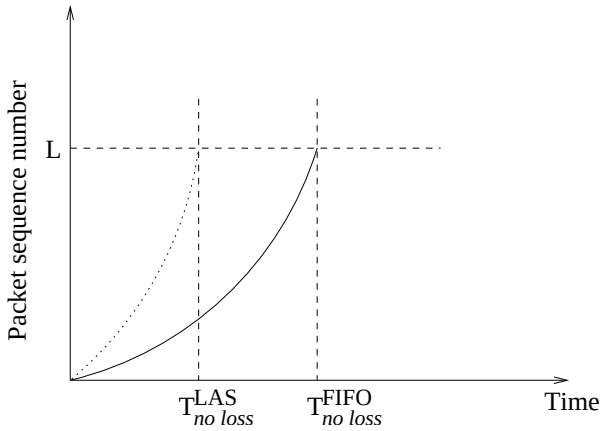


Figure 1: *SS behavior for short TCP flows with no packet loss under both polices.*

the transmission time T_{loss}^{FIFO} of the flow.

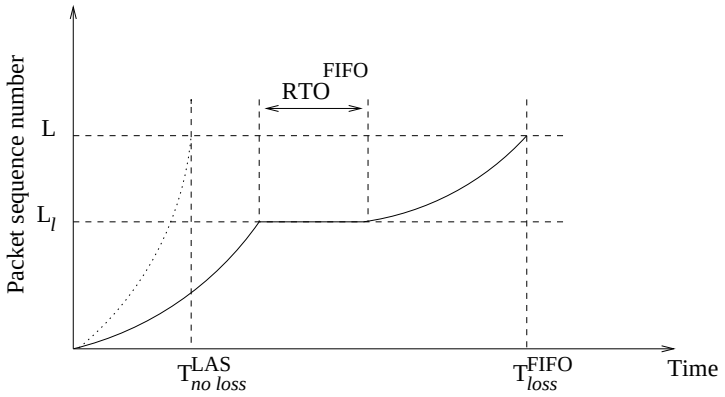


Figure 2: *SS behavior for short TCP flows with packet loss under FIFO.*

Consider also the case when the short flow under both policies losses the $(L_l + 1)$th packet. Figure 3 shows that the response time under LAS is still lower than the response time of the flow under FIFO. It is important to note here that RTO^{FIFO} is bigger than the RTO^{LAS}. This is because the computations of RTO in TCP at any time depends directly on previous RTT sample values [34], which are smaller under LAS than under FIFO. In practice, packet losses are likely to occur sooner under FIFO than under LAS. In this case, FIFO is highly likely not to have enough RTT samples to compute the RTO, and the TCP has to use a conservatively chosen *Initial Timeout* (ITO) value as RTO, which is often set to the high value of 3 seconds. This again lengthens the transmission time of short flows under FIFO. Given the same load conditions for both policies, it is unlikely that a short flow losses packets only under LAS but not under FIFO. So we do not elaborate this case. To summarize, we saw that the two main factors that make LAS improve the performance of short TCP flows are its inherent features of reducing the RTT during SS phase and reducing losses of first packets of each flow. Note that the arguments presented in this section applies to short flows but also to long TCP flows during their first SS phase. This means that LAS enables all flows to reach *congestion avoidance* (CA) phase faster than FIFO does.

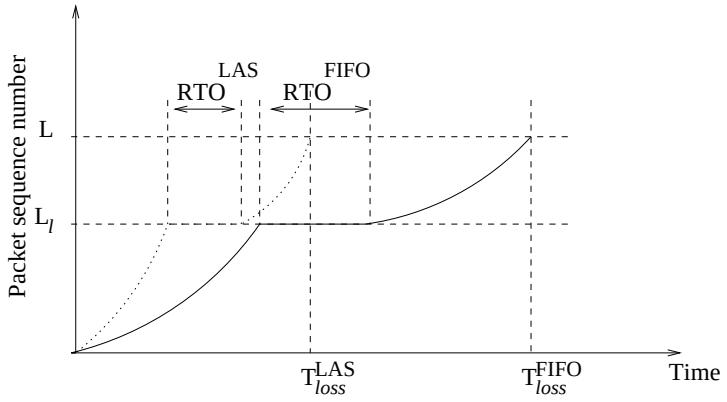
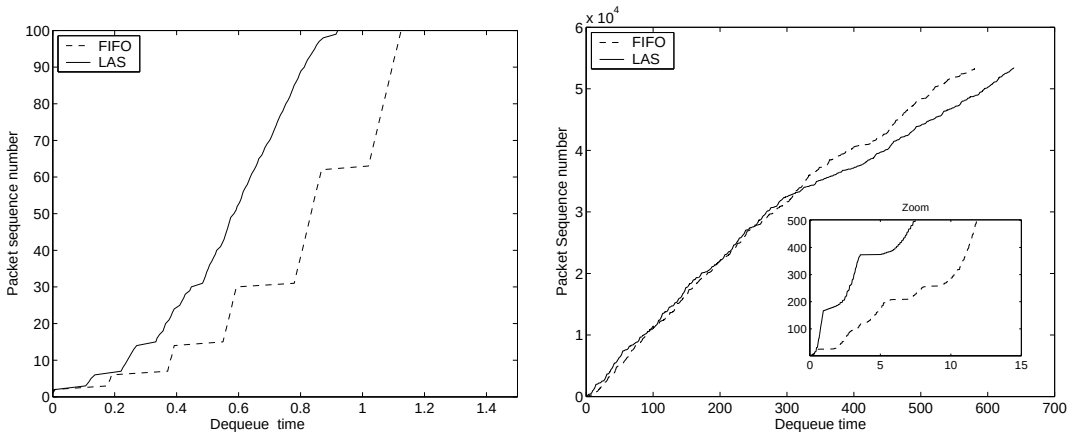


Figure 3: *SS behavior for short TCP flows with packet loss under both policies.*

3.2 Illustration of LAS at Packet Level

Now, we illustrate the performance of flows under LAS at packet level. These plots are derived from traces obtained by simulating LAS in network simulator (ns2) using Web traffic parameters and network topology as given in Section 4.1. For flows of equal size, we compare the absolute time each packet leaves the queue (*dequeue time*) when served under LAS and FIFO respectively assuming that the first packets in both cases leave the queue at the same time. The results nicely illustrate the impact of LAS, as compared to FIFO, on short and large TCP flows.



(a) *Web object of size 100 packets*

(b) *Web object of size 53,080 packets*

Figure 4: *Packet level analysis of LAS, load* $\rho = 0.7$

Figures 4(a) and 4(b) show the times when the packets of the flows leave LAS and FIFO as a function of packet sequence number for flows with 100 packets and for a single very large flow respectively. Figure 4(a) shows that all packets leave the queue under LAS sooner than under FIFO, which shows that LAS reduces the time that each packet of these flows spends in the queue. Figure 4(b) shows the same tendency up until 3500 packets of flows are sent, after that, packets under FIFO leave the queue sooner than under LAS. This elaborates the fact that under LAS the priority of a flow decreases with the increasing amount of data the flow has sent. Both figures illustrate how LAS speeds up the transfer of the first packets of all flows, which makes LAS a very efficient mechanism to improve the performance of short flows. We also observe from Figure 4(a) that LAS scheduler also has the advantage of smoothing

the bursts of TCP packets, which we see not to be the case for FIFO scheduler.

4 Simulation Results for Short Flows

4.1 Network topology and Web traffic model

To evaluate LAS, we use the network topology shown in Figure 5, where C1-C5 are clients initiating a series of Web sessions. During each session, a client requests several Web pages, each containing several objects from a randomly chosen server in the pool S1-S5 of servers. This Web model was first introduced in [17]. Each time an object is requested, a new TCP connection is established[3]. Thus the model emulates the HTTP 1.0 protocol. We refer to the packets that belong to one TCP connection as a **flow**.

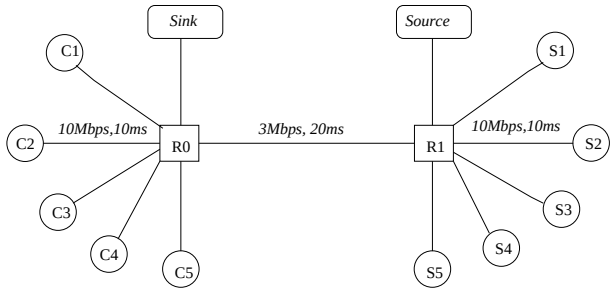


Figure 5: *Simulated network topology*

The simulated Web traffic profile is obtained by setting the distributions of inter-session, inter-page and inter-object time (all in seconds), session size (in web pages), page size (in objects) and object size (in packets) as seen in Table 1, which result to a total load for Web traffic of 0.7. We consider a Pareto *object size* distribution denoted as $P(a, \alpha, \bar{x})$, where a is the minimum possible object size, α is the shape parameter, and $\bar{x}$ is the mean object size. The Pareto distribution has high variance and models well flow size distributions as empirically observed in the Internet [21, 35, 14]. We also use $Exp(1/\mu)$ to denote exponential distribution with mean $1/\mu$. Our values shown in Table 1, are also used in [9], and are based on the findings of [38].

# of sessions	pages/session	objs/page	inter-session	inter-page	inter-obj	obj size
1300	*Exp(60)*	*Exp(3)*	*Exp(8)*	*Exp(5)*	*Exp(0.5)*	*P(1,1.2,12)*

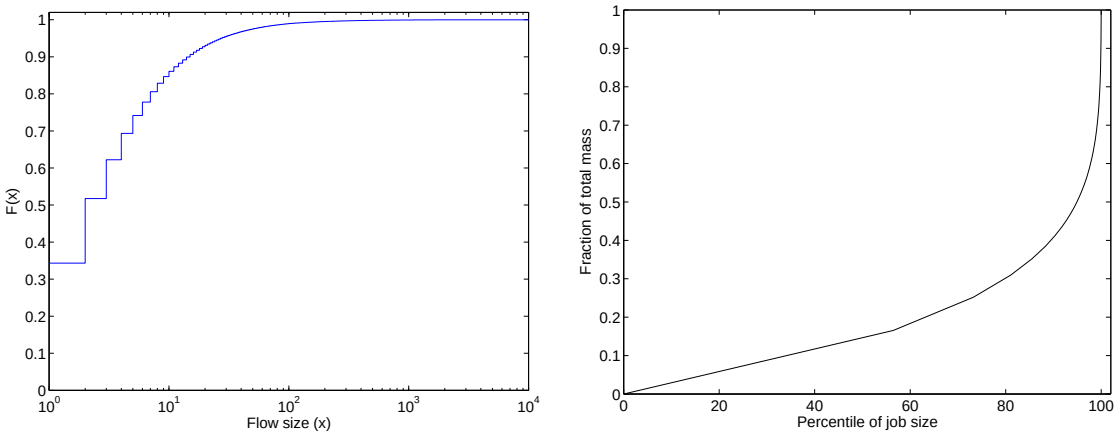
Table 1: Web traffic profile

Figure 6(a) shows the distribution of flow sizes obtained from the parameters shown in Table 1. We see that more than 80% of the flows have less than 10 packets and that flow sizes of more than 100 packets constitute less than 1% of all flows.

It is also useful to visualize the variability of a distribution in terms of its *mass-weighted distribution*. Denoting the cumulative distribution function of flow sizes as $F(x)$, the mass-weighted function is defined [13] as $F_w(x) \triangleq \frac{\int_{-\infty}^{x} udF(u)}{\int_{-\infty}^{\infty} vdF(v)}$. If plotted against $F(x)$, $F_w(x)$ yields the fraction of the total mass constituted by all flows whose size is less than or equal to x. Observe from Figure 6(b) that the 5% of

[3]For this reason, object size and flow size are the same, and we use both terms interchangeably.

the largest flows account for about 50% of total load for Pareto flow sizes Thus, the traffic we generate using the Pareto distribution in simulations represents the behavior of the Internet traffic observed today.



(a) *CDF of flow sizes*

(b) *Fraction of the total mass as a function of their cumulative distribution functions*

Figure 6: *Distribution of flow sizes for* $P(1, 1.2, 12)$ *and* $Exp(12)$ *distributions*

In Figure 5, *Source* is either an infinite TCP or UDP source sending data to the destination *Sink*. The bottleneck link in the network topology is R0–R1 with its queue size limited to 60 packets. We analyze and compare the performances of LAS and FIFO schedulers when placed in the bottleneck link. For all simulations, packets that carry Web traffic or long-lived FTP data have a size of 1040 bytes, and packets of the long-lived UDP flow are 1000 bytes in size.

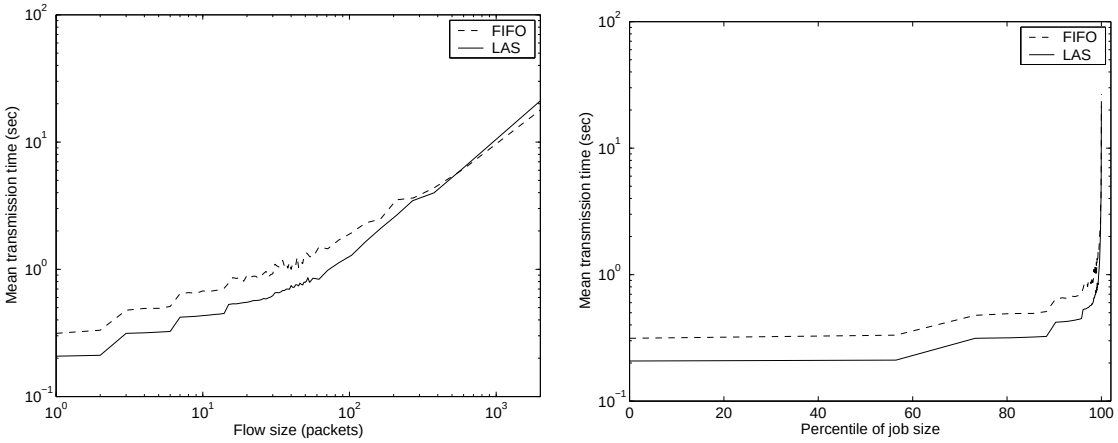
Each simulation is run for 12000 seconds, and the statistics are collected after a warm up period of 2000 seconds. Similarly, if *Source* is active, it will start sending data after 2000 seconds. In the next section, we discuss simulation results that compare the performance of LAS to FIFO in terms of mean response time and packet losses for short flows.

4.2 Mean transmission time

In this section, we discuss simulation results that show how LAS reduces the mean transmission time of short flows. Here, the transmission time of flow is the time interval starting when the first packet of a flow leaves a server and ending when the last packet of the flow is received by the corresponding client.

Figure 7(a) and 7(b) show the mean transmission time for LAS and FIFO as a function of object size and percentile[4] of object size respectively. It is evident from the figures that LAS significantly reduces the mean transmission time with a negligible penalty to large flows. We note that more than 99% of the short flows benefit from LAS. In particular, the mean transmission time of most of these short flows is reduced by a factor of about 2.5 under LAS. We also observe that the mean transmission time of large flows under LAS shows only a small increase compared to FIFO. It is worth recalling, however, that based on the distribution of generated flow sizes, these large flows (of size greater than 100 packets) constitute less than 1% of all flows.

[4]The pth percentile is the flow size x_p such that $F_x(x_p) = p/100$, where $F_x(x)$ is the cumulative distribution function of x.



(a) *Flow sizes* (b) *Percentiles*

Figure 7: *Mean transmission time of Pareto Web flow sizes, load* $\rho = 0.7$

Table 1 indicates that on average, there are three objects in a page and the average object size is 12 packets. This implies that the average page size is 36 packets, which is about 37KB. This number complies with the findings in [27] that the average size of all data on a Web page was 26-32KB. We see from Figures 7 that the mean response time of the Web transfer with 36 packets is lower under LAS than under FIFO. Thus, even if all objects in a page are transmitted using a single persistent HTTP connection, as the case for HTTP1.1, they will still experience a lower average transmission time under LAS than under FIFO.

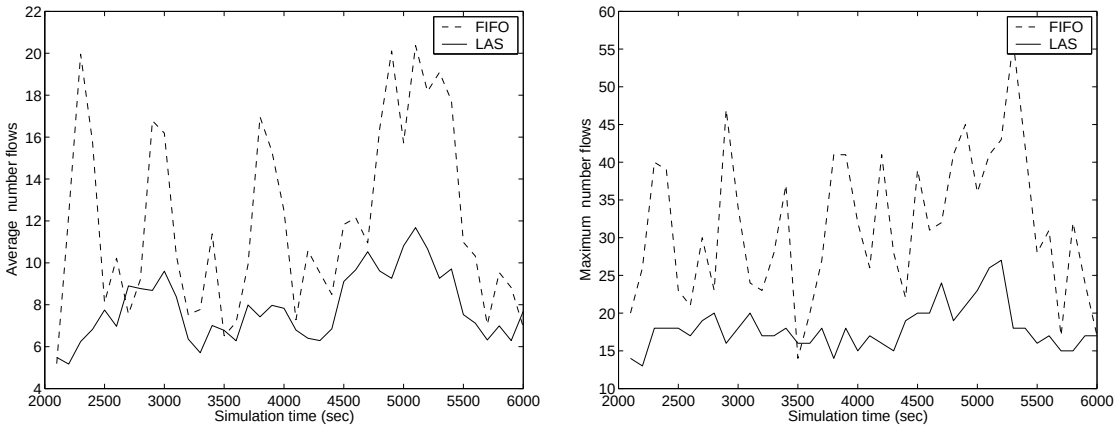
In this section, we presented the results only for a single load value of $\rho = 0.7$. We also looked at other loads and observed that short flows also benefit under LAS at lower load values like load $\rho = 0.5$ and the performance of short flows under LAS as compared to FIFO improves with increasing load values.

4.3 Number of simultaneous active flows

Little's law in queueing theory reveals the proportional dependency between the mean transmission time and the average number of jobs. Using the same definition for flows, here we claim that the reduction in the mean response time of flows under LAS as seen in Figure 7 should also result to the reduction in the average number of flows in the network. This is because each client generates the same distribution of short and long flows that results to a constant average arrival rate in steady state. Hence, the steady state throughput also becomes constant. Figure 8(a) shows the average number of active Web flows in the network topology seen in Figure 5 for the cases when the bottleneck router uses LAS and when it uses FIFO. The figure clearly shows that the average number of flows under LAS is lower than under FIFO, which claim that it is the result of Little's law for the case of LAS since it also reduces the mean transmission time.

Figure 8(b), shows the maximum number of flows in the network for both, LAS and FIFO schedulers, in every non-overlapping time intervals of 100 seconds. The results show the significant difference between the two policies; the maximum number of active flows under FIFO is close to twice the maximum number of active flows under LAS. In both plots, the small difference value between the policies for simulation time less than 4000 seconds only indicates that the simulation is still in transient state during that time.

These results also show that LAS scheduler actually needs to maintain a smaller number of simultaneous



(a) *Mean number of flows* (b) *Maximum number of flows*

Figure 8: *Number of flows as a function of simulation time, load* $\rho = 0.7$

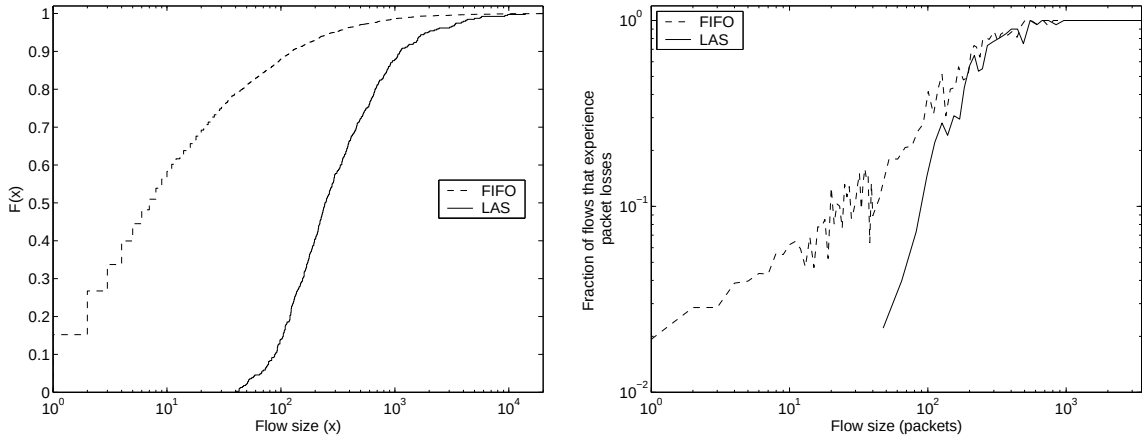
active flows in the system than the number of flows observed under FIFO for the same traffic. This is important when implementing LAS in routers (discussed in Section 6) as the number of flows to keep state information in LAS routers is reduced by the properties of the scheduler itself.

4.4 Loss rate

The overall performance of TCP depends to a certain degree on how it can efficiently detect and reduce network congestion. In general, short TCP connections are not capable of controlling congestion. When a short flow with very few packets (e.g., 1, 2, or 3 packets) experiences loss(es), the congestion window is too small that reducing its size will have any impact on reducing congestion. On the other hand, the network congestion is alleviated when a source with a large congestion window detects loss. In this case, reducing the congestion window reduces the overall network load and therefore the congestion. Hence, reducing the congestion windows of a few flows with large congestion windows can be sufficient to react to network congestion.

LAS speeds up the SS phase of all flows (including largest ones) and enables sources to reach the CA phase faster. This improves the throughput performance of TCP since TCP flows under LAS, especially large flows, are likely to detect packet losses during the CA phase. On the other hand, in routers with FIFO scheduling, short flows are equally likely as large flows to see their packets dropped, which deteriorates their throughput performance without helping to alleviate congestion. When the queue is full, LAS first inserts an incoming packet at its position in the queue (based on its priority value), and then drops the packet that is at the tail of the queue. It turns out that LAS very much protects flows from losing packets during the initial SS phase and most likely drops packets from large flows. Apparently, simulation results also assert that speeding up the SS phase and avoiding packet losses in this phase make LAS such an effective policy for short flows. In this section, we discuss the simulation results that illustrate the positive impact of LAS on the packet losses seen by short flows.

The simulation is performed with parameters of Table 1 for Pareto distributed flow sizes. The overall loss rate for LAS is about 3.3% whereas for FIFO it is about 3.9%. The number of flows that experience one or more packet loss under FIFO is 12212, almost twice the number of flows that experience loss under LAS, which is 6763. Figure 9(a) shows the distribution of flows that experience packet losses under LAS and FIFO. We observe that the percentage of short flows that experience losses under LAS is much



(a) *CDF of flows that experience loss* (b) *Flows loss ratio*

Figure 9: *Loss analysis for Pareto distributed Web flow sizes, load* $\rho = 0.7$

smaller than the percentage of short flows that experience loss under FIFO. Figure 9(a) shows that about 18% (or 2198 flows) of the flows that experience packet loss under FIFO are of size one packet whereas under LAS only 11 flows of size one packet experience loss. The difference is even higher for flows with slightly more than one packet; flows with less than three packets make up only 8% of all flows that experience losses under LAS, a much smaller fraction than under FIFO, where it is 30%. Considering flows with less than 10 packets, we see that they make 70% of flows that experience losses under FIFO, which is more than twice the percentage seen under LAS. In general, we observe that it is essentially the short flows that benefit from LAS in terms of significantly reduced packet loss. Furthermore, Figure 9(b) shows the ratio of flows that experience packet losses over all flows of a given size. Again, we note a significant difference between LAS and FIFO for short flows; for example, under FIFO about 4% of flows of size one suffer a loss compared to only about 0.015% under LAS.

Finally, it is also useful to analyze the average packet loss rate to understand if LAS reduces the loss rate of short flows by increasing the loss rate for large flows. Figure 10 shows the average packet loss rate as a function of flow size. It is evident from that figure that LAS reduces the loss rate for short flows and

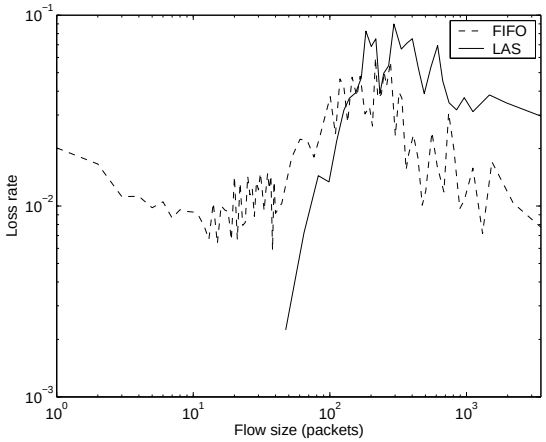


Figure 10: *Packet loss rate for Pareto distributed Web flow sizes,* $\rho = 0.7$

at the same time maintains for large flows loss rates similar to the ones seen under FIFO. Therefore, the

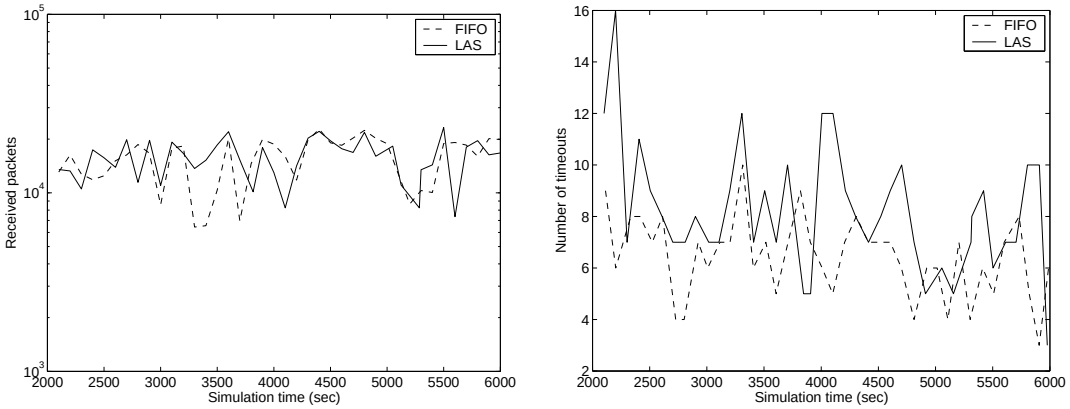
significant reduction of loss rate of short flows under LAS does not come at the expense of a higher loss rate for large flows as compared to FIFO. The zoom of the figure elaborates the loss rates for short flows of sizes less than 10 packets on a linear scale. In summary, it seems that the substantial packet losses reduction offered by LAS for short flows is the most important factor that improves the performance of short TCP flows under LAS as compared to under FIFO.

5 Single Long-lived Flow Competing with Web Background Traffic

We can see from Figure 7 that the mean transmission times of the largest generated Web flows under LAS do not differ much from their mean transmission times under FIFO. Similarly, we observed in Section 4.4 that the loss rates of large Web flows under LAS and FIFO are comparable. These results allow us to conclude that the performance of LAS for short flows comes with a negligible penalty in terms of increase of loss rate and mean transmission time of large flows. In this section, we want to investigate the performance of even larger (long-lived) TCP and UDP flows than the largest transmitted Web flow under LAS and compare the results under FIFO.

5.1 Single long-lived FTP flow

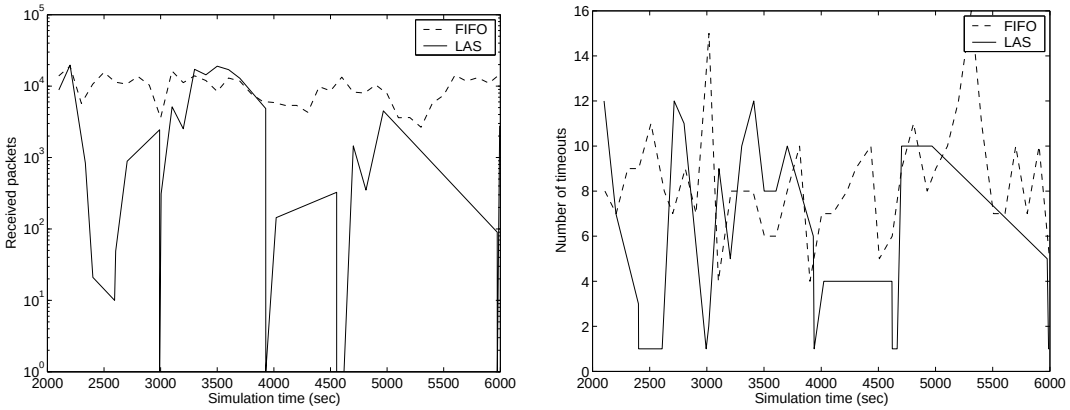
We use again the network topology shown in Figure 5. The long-lived FTP flow is represented by *Source* that starts sending data packets to *Sink* at 2000 simulation seconds. The FTP source uses TCP Reno. The background Web traffic is produced using the parameters from Table 1, with Pareto distributed Web flow sizes. We consider three load conditions: a total load of $\rho = 0.75$, $\rho = 0.9$, and $\rho = 0.98$. At load $\rho = 0.9$, the background Web traffic is generated using parameters shown in Table 1, whereas to obtain a total load of $\rho = 0.75$ and $\rho = 0.98$, we adjust the distribution of inter-session time in Table 1 to $Exp(11)$ and $Exp(6)$ respectively. We are interested in timeout and throughput behavior of a large FTP transfer that shares the bottleneck link with the Web traffic.



(a) *Throughput* (b) *Timeouts*

Figure 11: *FTP flow at load* $\rho = 0.75$

Intuitively, one expects the FTP source under LAS to starve as a result of LAS favoring short TCP flows. The simulation results show that the starvation does not occur except at very high load values of $\rho > 0.9$. Figures 11 and 12 show the number of timeouts and the number of packets received in non-overlapping time intervals of 100 seconds as a function of simulation time for load $\rho = 0.75$ and $\rho = 0.9$ respectively.

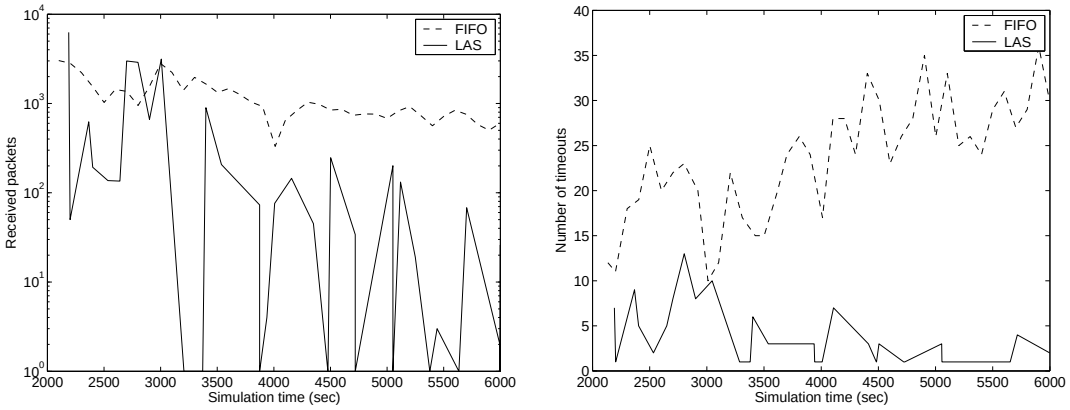


(a) *Throughput* (b) *Timeouts*

Figure 12: *FTP flow at load* $\rho = 0.9$

For the case of $\rho = 0.9$, we observe only a few time instances when the long-lived flow is completely starved. These are possibly the epochs when the generated Web short flows have high transient load.

The situation is more different for a very high total load of $\rho = 0.98$, where LAS is observed to starve the FTP flow after some time instance (Figure 13). For LAS, both the number of timeouts and the throughput widely oscillate and decrease on the average over time. During some periods, throughput value drops all the way to zero, when the FTP flow completely shuts off as the underlying TCP source stops sending packets for some time. Note that this occurs after a notable simulation duration of 3000 seconds, which is equivalent of about 10^3 packets or about 1 Mbytes of data transfer of the FTP flow . Recal that the bottleneck link at high load gets very congested and LAS starves the long-lived flow as a result of favoring many short ones. Under FIFO however, starvation does not occur and the FTP source continues sending the data until the simulation terminates. We observe from the figure for the case of FIFO that the number of timeouts only slightly increases and the throughput of the flow slightly decreases as the simulation time increases. FIFO however does not favor short flows, and this result is consistent with the well known lock-out phenomenon of long-lived TCP flows against short flows under FIFO.

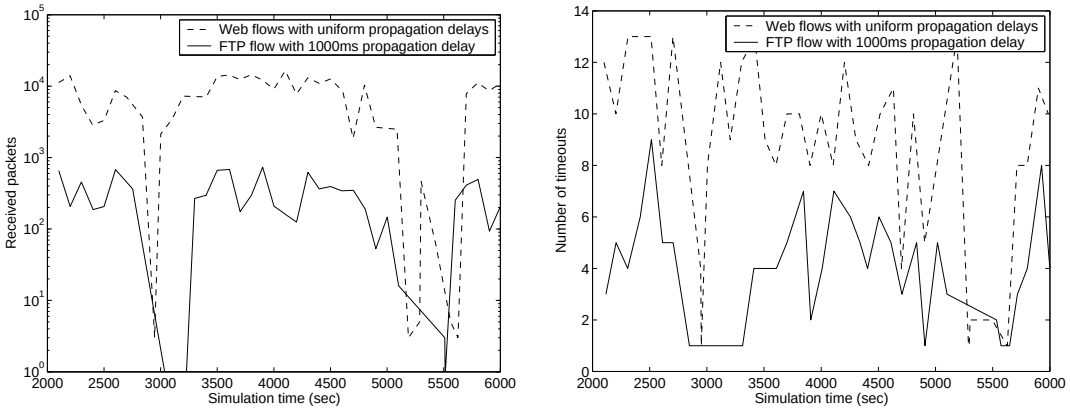


(a) *Throughput* (b) *Timeouts*

Figure 13: *Performance of FTP flow at load* $\rho = 0.98$

5.2 Impact of varying propagation delays

In the previous analysis, we considered the topology when all servers-client pairs have the same one way propagation delay of 50ms. When flows have the same RTT values and maximum advertized window values, they also have equal maximum source rates. This increases the burstness of traffic at the bottleneck and under LAS may cause the performance of a long-lived flow to deteriorate. In this section, we show simulation results when the propagation delays in Figure 5 are changed.



(a) *Throughput* (b) *Timeouts*

Figure 14: *Performance of FTP flow for topology links with varying propagation delays*

We consider tow cases: the case when only the propagation delay of the FTP source is changed to 1000 seconds and the case when propagation delays of links connecting servers and clients are uniformly distributed between 50ms and 400ms. Web parameters are kept the same as those used in Section 5.1 and resulted to total load of $\rho = 0.9$. Simulation results showing the performance of the FTP flow for both cases are shown in Figure 14. We observe that compared to the results shown in Section 5.1, LAS offers higher throughput when propagation delays of links are not the equal. Note that the throughput is higher when propagation delays are uniformly distributed than when only the propagation delay of the FTP flow is increased. We also observed that the number of timeouts are also higher for the case of uniformly distributed propagation delays but in most cases they are higher than 1, which show that the FTP flow hardly completely shuts off.

5.3 Single long-lived UDP flow

A recent study showed that only about one third of todays media streams in the Internet are carried using UDP as transport protocol [39]. When FIFO scheduling is used in routers, open-loop traffic can pose a danger to TCP traffic because it does not respond to network congestion as do TCP-based applications. It is well known that when UDP-based applications share the bottleneck link with TCP-based applications in a FIFO router, they tend to grab as much bandwidth as corresponds to their source rates, while TCP flows will back-off and see their throughput decreased. LAS, by definition, prevents any flow to consume a larger share of bandwidth than any competing flow, regardless of the underlined transport protocol. In particular, LAS fairly services all flows that have received the same amount of service in the system.

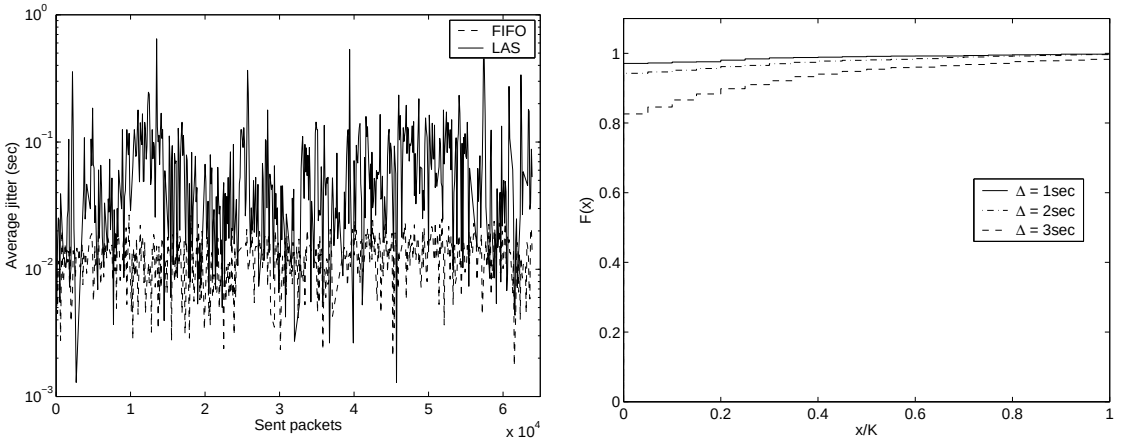
Here we only consider long-lived UDP flows but it is obvious to see that short UDP flows can benefit form LAS more than short TCP flows; for short UDP and TCP flows of equal sizes that arrive at the LAS router at the same time, the UDP flow (particularly with CBR source rate) can complete the service

earlier than the TCP flow that is subject to closed loop control mechanisms. In the next section, we investigate the performance of a very large UDP flow when competing with Web traffic under the LAS scheduling policy.

A Jitter analysis

We use the network topology shown in Figure 5 where *Source* is a UDP source that starts sending data to *Sink* at a Constant Bit Rate (CBR) of 16 packets/sec after of 2000 seconds. Thus, the UDP source generates a long-lived flow of $16 * 10^4$ packets, which makes 160 Mbytes of data. The Web background traffic is generated using the parameters shown in Table 1, with Pareto distributed Web flow sizes.

The jitter performance is an important performance metric for multimedia applications. Figure 15(a) shows the jitter averaged over non-overlapping groups of 100 sent packets. It is evident that the UDP flow experiences a much higher average jitter under LAS than under FIFO. The UDP flow under LAS is always pre-empted by new TCP flows, which delays its packets in the system and so increases the jitter.



(a) *Average jitter*

(b) *The CDF of the ratio of the packets in a block of* $K = 20$ *packets that miss deadline*

Figure 15: *UDP flow performance, load* $\rho = 0.75$

While the simulation results show high jitter behavior for the long-lived UDP flow under LAS, we argue that a small *playout buffer* of not more than 4 seconds for this simulation scenario guarantees that no packet misses the *deadline*. A packet is said to miss the deadline if the time difference between its send time t_s and the receive time t_r is larger than or equal to the sum of the playout buffer size Δ and the average transmission time $\bar{T}$; i.e., $t_r - t_s > \Delta + \bar{T}$. Figure 15(b) shows the cumulative distribution of the ratio of the packets in a block of $K = 20$ packets that miss deadline, where $\bar{T} = 0.5$ seconds. We observe from the figure that the number of packets that do not miss their deadlines increases for increasing playout buffer size. For $\Delta = 3$ seconds about 98% of blocks have all packets arrive at the destination in time, and all packets arrive in time for $\Delta = 4$ seconds.

B Loss analysis of a long-lived UDP flow

Figure 16 shows the number of packets lost in every 100 packets sent. Under LAS, the long-lived UDP flow has a lower priority than (most of) Web background traffic. As the UDP flow is open-loop and does

not adapt its rate, the UDP flow can suffer a significant loss during periods of congestion. Overall, the average loss rate of the UDP flow under LAS is 0.0118, which is about 60% higher than the value of 0.0005 observed under FIFO.

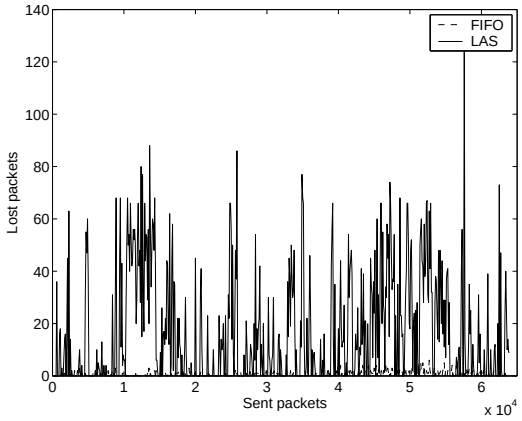


Figure 16: *Loss rate for a UDP flow at load* $\rho = 0.75$

The scheduling policy has a big impact on how closed-loop TCP flows and open-loop UDP flows interact in a bottleneck router; (i) Under LAS scheduler, the TCP flows of Web traffic have buffer space and service priority over the long-lived UDP flow. As a consequence, in case of congestion, the long-lived UDP flow suffers high jitter and an increased loss rate. (ii) Under FIFO scheduler it is the opposite as UDP flows will degrade the performance of the TCP flows: As the open-loop UDP flow does not react to congestion, it will be the TCP flows that suffer loss and back-off, seeing their share of the bottleneck bandwidth reduced.

6 Implementation Issues

Implementing LAS scheduler involves two main tasks: Maintaining a priority queue that implements the highes-priority-first queue, and keeping state of all active flows in the network at any time. There are a number of methods proposed to implement a priority queue for high speed packet networks. Moon et. al. present a detailed comparison of different existing hardware approaches used to build priority queues [29]. The work in [7] presents a hardware implementation called *pipelined priority queue architecture* for a priority queue that implements the highest-priority-first schedulers (like LAS). The architecture presented in [7] is shown to be fast and to scale well with respect to the number of priority levels and the queue size. In particular, the work shows that the implementation can support connections with up to 10Gb/s rates and over 4 billion priority levels.

In this paper, we propose a *priority assignment unit* in a LAS router whose functions include computing the priority of each arriving packet and inserting the packet to its appropriate position in the queue. The priority assignment unit then needs to track for each flow f the amount of bytes S_f served so far. S_f is initialized with 0. The priority value S_f is the priority of the packet and so is used to determine where to insert the next packet of flow f into the priority queue. Recall that the lower the priority value the higher the priority, and the closer to the head of the queue the packet is inserted. When a packet of size P for flow f arrives, the priority assignment unit executes the following operations:

– Use S_f to insert packet into priority queue
– Update S_f value to $S_f := S_f + P$

LAS in a router will be non-preemptive and the packet to be served next will be taken from the head of the queue.

Keeping per-flow state is known to be a difficult task in the core routers due to the presence of a large number of simultaneous active flows at the network core. However, a recent work on network tomography [32] showed that Internet congestion (packet losses) mostly occur at the edge and access links of the Internet, where the number of flows to keep state is moderate and so manageable by LAS routers. Thus, deploying LAS in the access and edge routers should be sufficient to reap the benefits of LAS in terms of reducing the response time of short TCP flows.

In case LAS needs to be implemented in all routers of the network, the Diffserv like model, which pushes per-flow state complexity to edge routers, can be adopted. In this case, priority assignment units in edge routers compute and stamp priority value of each each packet in a priority field to be created in packets headers (See [7] for possible parameters of a packet priority field). Core routers then do not need to compute priorities and to keep per-flow state. Instead, they use the priority field value of each incoming packet to insert the packet into its appropriate position in the queue.

7 Conclusion and Open Issues

LAS scheduling for jobs has been known for over thirty years. This paper is a first attempt to evaluate the impact of LAS scheduling in a router to TCP and UDP flows. Using a realistic traffic scenario that captures the high variation of the flow sizes observed in the Internet, we saw that LAS scheduler in the bottleneck router (i) significantly reduces the mean transmission time and loss rate of short TCP flows, (ii) does not starve long-lived TCP flows except for load values close to or corresponding to overload. Since LAS gives space and service priority to newly arriving flows, it degrades the jitter and loss performance of long-lived UDP flows. However, a small playout buffer at the destination allows to compensate for the jitter.

Various authors (see Section 2) have proposed to improve the performance of short TCP flows by changing the mechanisms deployed in the routers. However, our proposal has several distinctive features

- LAS is conceptually very simple as there is not a single parameter that needs to be set
- The priority queue of LAS improves the transmission time of short flows much more than schemes that propose to use *separate FIFO* queues for short and long flows
- LAS protects TCP flows against open-loop UDP flows, which FIFO scheduling does not

Perhaps the most important issue to explore is to determine *where* in the network to deploy LAS. We are aware that the deployment of a new scheduling policy in routers will face many obstacles and wide deployment will be an elusive goal. However, we expect that a limited deployment of LAS in certain routers at well identified bottlenecks and where the number of flows to keep information is scalable allows to reap most of the benefits. As we have seen, such bottlenecks can be the access links, edge links, or the transition from the wired to the wireless Internet.

There remain a number of questions regarding the performance benefits of LAS for short and long TCP flows. The interaction between LAS and TCP's error and congestion control is quite intriguing. It is worthwhile to better understand which of the two factors, the reduction in RTT or the reduction in loss rate for the first packets in a flow, contributes most to the reduction in transmission time. FIFO is known to be unfair against some flows in heterogeneous networks environments. Examples of these networks

include networks shared by UDP and TCP applications, asymmetric networks with paths of varying propagation delays, and TCP flows that traverse multiple congested gateways. By definition, the LAS scheduler should enforce fair bandwidth sharing among competing flows since LAS gives service to flows based on the amount of data sent. Hence, intuitively no flow under LAS, regardless of its transport protocol, can hog the link bandwidth and lock out other flows. This is a problem that we are currently analyzing.

References

[1] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing", In *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[2] M. Allman et al., "An Evaluation of TCP with Larger Initial Windows", *ACM Computer Communication Review*, 28(3):41–52, July 1998.

[3] M. Allman et al., "Increasing TCP's Initial Window", Request for Comments RFC 2414, Internet Engineering Task Force, September 1998.

[4] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, Internet Engineering Task Force, April 1999.

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, S. M., and R. Katz, "TCP Behavior of a Busy Internet Server: Analysis and Improvements", In *Proc. Infocom 98*.

[6] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1992.

[7] R. Bhagwan and B. Lin, "Fast and Scalable Priority Queue Architecture for High-Speed Network Switches", In *INFOCOM 2000*, pp. 538–547, 2000.

[8] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance", In *Proceedings of the ACM SIGCOMM Conference*, London, England, 1994.

[9] X. Chen and J. Heidemann, "Preferential Treatment for Short Flows to Reduce Web Latency", *To appear in Computer Networks*, 2003.

[10] M. G. Claffy, K. and K. Thompson, "The nature of the beast: Recent traffic measurements from an Internet backbone", In *Proceedings of INET '98, July 1998*, July 1998.

[11] E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall Inc., 1973.

[12] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley Publishing Company, 1967.

[13] M. E. Crovella, "Performance Evaluation with Heavy Tailed Distributions", In *Job Scheduling Strategies for Parallel Processing 2001 (JSSPP)*, pp. 1–10, 2001.

[14] M. E. Crovella et al., *A Practical Guide to Heavy Tails*, chapter 3, Chapman and Hall, New-York, 1998.

[15] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", In *Proc. of ACM SIGCOMM'89*, pp. 1–12, Austin, Texas, September 1989.

[16] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, 26(3):5–21, June 1996.

[17] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control", In *Proc. of the ACM/SIGCOMM'99*, August 1999.

[18] L. Guo and I. Matta, "The War between Mice and Elephants", In *Proc. 9th IEEE International Conference on Network Protocols (ICNP'01)*, Riverside, CA, 2001.

[19] L. Guo and I. Matta, "Differentiated Control of Web Traffic: A Numerical Analysis", In *SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, August 2002.

[20] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based Scheduling to Improve Web Performance", *ACM Transaction on Computer Systems*, (May), 2003.

[21] M. Harchol-Balter, "The Effect of Heavy-Tailed Job Size. Distributions on Computer System Design", In *Proc. of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics*, June 1999.

[22] J. Hoe, "Improving Start-up Behavior of a Congestion Svheme for TCP", In *ACM SIGCOMM*, pp. 270–280, 1996.

[23] http://www.isi.edu/nsnam/ns/, "The Network Simulator ns2",

[24] J. Ke, "Towards a Rate-based TCP Protocol for the Web", In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '00*, pp. 36–45, October 2000.

[25] S. Keshav, *An Engineering Approach to Computer Networking*, Prentice Hall, 1st edition, 1997.

[26] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*, Wiley, New York, 1976.

[27] B. A. Mah, "An Emerical model for HTTP network traffic", In *IEEE INFOCOM 1997*, pp. 592–600, April 1997.

[28] I. Matta and L. Guo, "Differentiated Predictive Fair Service for TCP Flows", In *Proc. 8th IEEE International Conference on Network Protocols (ICNP'00)*, pp. 49–57, Osaka, Japan, 2000.

[29] S. Moon, J. Rexford, and K. G. Shin, "Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches", *IEEE Transactions on Computers*, 49(11):1215–1227, November 2000.

[30] J. Nagle, "On packet switches with infinite storage", *IEEE Transactions on Communications*, COM-35(4):435–438, April 1987.

[31] V. Padmanabhan and R. Katz, "TCP Fast Start: a Technique for Speeding Up Web Transfers", In *Globecom'98 Internet Mini-Conference*, November 1998.

[32] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based Inference of Internet Link Lossiness", In *IEEE INFOCOM 2003*, 2003.

[33] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approch to Flow Control In Intergated Services Networks-The Single Node Case", In *Proceedings of Infocom*, pp. 914–924, IEEE, 1992.

[34] V. Paxon and M. Allman, "Computing TCP's Retransmission Timer", Request for Comments RFC 2988, Internet Engineering Task Force, November 2000.

[35] V. Paxon and S. Floyd, "Wide-Area Traffic: The failure of Poisson Modelling", *IEEE/ACM Transactions on Networking*, 3:226–244, June 1995.

[36] I. A. Rai, G. Urvoy-Keller, and E. W. Biersack, "Analysis of LAS Scheduling for Job Size Distributions with High Variance", In *ACM SIGMETRICS 2003*, San Diego, California, June 2003.

[37] L. E. Schrage, "A Proof of the Optimality of the Shortest Remaining Service Time Discipline", *Operations Research*, 16:670–690, 1968.

[38] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP protocol headers can tell us about the web", In *Proc. ACM SIGMETRICS*, pp. 245–256, June 2001.

[39] J. van der Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact", In *Proc. of the Seventh International Web Content Caching and Distribution Workshop*, August 2002.

[40] C. Williamson and Q. Wu, "A Case for Context-Aware TCP/IP", *Performance Evaluation Review*, 29(4):11–23, March 2002.

[41] S. Yilmaz and I. Matta, "On Class-Based Isolation of UDP, Short-Lived and Long-Lived TCP Flows", In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '01*, August 2001.

[42] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks", *IEEE Proceedings*, October 1995.

[43] Y. Zhang et al., "Speeding Up short Data Transfers", In *NOSSDAV 2000*, June 2000.