Self-Prioritization of Audio and Video Traffic

Thomas Bonald, Luca Muscariello, Norberto Ostallo

Self-Prioritization of Audio and Video Traffic

Thomas Bonald*, Luca Muscariello† and Norberto Ostallo‡
*Telecom ParisTech, France, thomas.bonald@telecom-paristech.fr
† Orange Labs, France Telecom, luca.muscariello@orange-ftgroup.com
‡ Eurecom, France, norberto.ostallo@gmail.com

***Abstract*—We present a packet scheduler called "shortest queue first" (SQF) that aims at protecting audio and video traffic from the congestion caused by data traffic. Unlike standard solutions, the services to be handled with priority are *not* known in advance. It is rather the traffic characteristics of audio and video applications that are used to detect their delay sensitivity. The SQF algorithm does not require any prior configuration of the network and, as such, adapts to the fast evolution of traffic and usage. The performance of the proposed solution is demonstrated using both analysis and experiments on a testbed emulating a residential access line.**

I. Introduction

We have faced a dramatic increase of video traffic in the last decade, exemplified by the popularity of video file sharing systems and the advent of TV over IP. This fast growth is expected to continue within the next few years, sustained by the development of video-on-demand services, catch-up TV, and P2P streaming applications. Such delay-sensitive services often suffer from the congestion caused by data traffic, especially on the 'last mile", namely the user's access line and possibly the first aggregation links of the backhaul network, as illustrated by Fig.1. In this paper we focus on streaming applications that require expedited packet forwarding like interactive communications (video-conferencing, VOIP) or Web-TV. Many other streaming applications, less sensitive to timely delivery, frequently make use of progressive downloads (VoD, YouTube, Daily Motions etc.) and are more robust to packet delays and losses at the cost of larger play-out delays. In this paper we do not consider such applications that rather require a minimum available bandwidth to the source, than expedited packet forwarding.

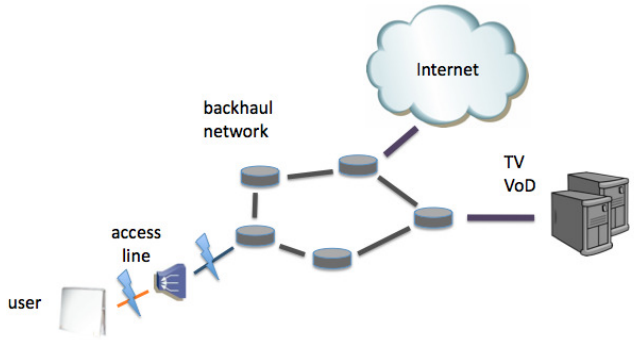


Fig. 1. Congestion on the network's last mile.

The current solution proposed by IETF and implemented by most suppliers relies on the Diffserv architecture [6]. Packets are marked at the network edge according to some pre-defined policy and handled with high or low priority in the network depending on their mark. This architecture is used by ISPs to protect their own telephony and video services for instance. It is hardly applicable to tier-services, however, that appear and evolve very rapidly according to the users' needs and the business opportunities. Changing the marking policy at this pace is usually considered as either infeasible or economically nonviable.

We propose a packet scheduler called "shortest queue first" (SQF) that introduces *implicit* service differentiation. The services to be handled with priority are *not* known in advance. It is rather the characteristics of audio and video applications, namely the fact that packet arrivals are regulated by some source codec and not by the congestion control mechanisms of TCP, that are used to detect their delay sensitivity. Note in particular that SQF equally applies to audio and video traffic over UDP *and* TCP, the congestion control mechanisms of TCP being typically inactive in the latter case (they become active only if packets queue in the network, which is precisely what SQF aims at avoiding). The algorithm does not require any prior configuration of the network and, as such, adapts to the fast evolution of traffic and usage.

The rest of the paper is organized as follows. Some related work is presented in the next section. The SQF algorithm is described in Section III. Sections IV and V give the performance results, obtained both by analysis and experiments on a testbed, when the SQF algorithm is activated on the user's access line. Section VI concludes the paper.

II. Related work

The SQF algorithm is somewhat related to fair queuing (FQ), a scheduling policy introduced by Nagle [18] as a means to protect TCP flows from non-responsive, UDP flows. Shreedhar and Varghese proposed an efficient implementation of FQ through the deficit round-robin (DRR) algorithm [23]. Some authors observed that FQ is not only desirable for fairness but also for the packet delay of those flows that use less than their fair share, see [9], [24]. This idea is exploited by Kortebi, Oueslati and Roberts, who first coined the term of "implicit" service differentiation [16]: their priority fair queuing (PFQ) algorithm gives strict priority to the packets of non-backlogged flows, which minimizes their packet delay;

backlogged flows are scheduled according to the standard FQ policy.

Unlike FQ and PFQ, SQF is *not* based on the throughput of active flows but only on the number of packets they have in the buffer. Specifically, SQF always serves a packet of that flow having the smallest amount of data in the buffer, hence the name. It is thus the traffic characteristics of audio and video flows, and in particular the fact that packet arrivals are regulated by the source codec, that are used to schedule them with high priority. In particular, SQF is able to prioritize audio and video traffic of high rates, close to the link capacity.

In [8], SQF has been analyzed in a link fed by multiple TCP sources, through fluid differential equations. In addition, the performance of CBR sources in presence of TCP traffic has been considered. A main limitation of fluid modeling is that packet fluctuations are neglected, and does allow to extend the analysis in [8] to a larger class of sources.

In this paper we further analyze SQF, considering packet fluctuations to highlight some properties of the scheduler that cannot be measured using fluid modeling. We here present both analytical and experimental results comparing the performance of SQF and FQ in the more realistic context of VBR (variable bit rate) traffic.

The efficiency of SQF has also be demonstrated in [7] on the Livebox, an ADSL router available to customers of Orange's Broadband services[1].

It is worth observing that SQF is the exact opposite of the "longest queue first" (LQF) policy considered for instance by Dimakis and Walrand [11] as an approximate, distributed implementation of the maximal weight matching algorithm of Tassiulas and Ephremides [25]. These scheduling policies aim at maximizing the utilization of some interferring resources in the context of wireless networks [22] or packet switches [17] for instance. Our problem is very different: we focus on a single resource (e.g. the user's access line) and seek to optimize the multiplexing of different services (audio, video and data).

Finally, the drop policy plays a critical role in bandwidth sharing. Some proposals use FIFO scheduling and thus rely entirely on the buffer management to enforce fairness, see [20], [19]. As pointed out by Suter et. al. [24], FQ must also be combined with the Longest Queue Drop (LQD) policy to protect flows of low throughput. Similarly, we apply LQD to protect regulated flows: the combination of SQF and LQD guarantees both low delays and low loss rates to audio and video traffic, as shown in the rest of the paper.

III. Shortest Queue First

A. Flow identification

Like FQ and PFQ, the SQF algorithm is "flow-aware": each packet is dropped or served depending on the flow it belongs to. For present purposes, we refer to a *flow* as the set of packets that belong to the same instance of application. In practice, a flow is identified either through the 5-tuple of the IP header, or through some pre-defined part of it. The exact definition depends on the desired degree of traffic aggregation. On the *upstream* of the user's access line for instance, that is in the equipment connecting the user to the network, it may be beneficial to consider the *source port* as the flow identifier so as to aggregate all packets of the same application, whatever the destination of these packets. All packets sent to the various peers of a P2P file swarm are then considered as belonging to the same flow, for instance.

[1] http://en.wikipedia.org/wiki/Orange_Livebox

B. Packet scheduler

A flow is said to be *active* if it has at least one packet in the buffer. The packet scheduler maintains one virtual queue per active flow and always serves a packet from the shortest queue in bytes. It is thus the *volume* of data of each flow that is used to select the shortest queue, making the algorithm insensitive to the packet size distribution. The algorithm is work-conserving and non-preemptive. Note that the number of active flows is upper bounded by the buffer size, which makes the scheduler perfectly scalable.

Algorithm 1 Shortest Queue First

```
At packet pkt arrival:
   queue[pkt.flowid]->enqueue(pkt);
   if (queue->size > queue->maxsize) {
      flow_id_longest = get_longest(queue);
      queue[flow_id_longest]->drop();
   }
At packet departure:
   flow_id_shortest = get_shortest(queue);
   queue[flow_id_shortest]->dequeue();
```

As mentioned above, the SQF algorithm is combined with LQD so that regulated flows (with low virtual queues) do not suffer any packet drop, as illustrated by Fig. 2 and the above pseudo-code. The total buffer is then simply sized so as to ensure an efficient utilization by TCP; typical values range from 50 to 200 MTU sized packets, see [21].

(a) Packet arrival (b) Packet departure

Fig. 2. Shortest Queue First.

IV. Analytical results

This section is devoted to the performance analysis of SQF. Specifically, we assess the robustness of SQF to the jitter of one or several VBR video flows in the presence of data traffic, for which the considered link is assumed to be the bottleneck.

A. A single video stream

We first consider a single video stream that competes for bandwidth with n TCP flows. We assume that these TCP flows saturate the buffer and have approximately the same number of packets in it. For simplicity, we also assume that all video packets have the same size, taken as the data unit.

Let x be the number of video packets in the buffer. Denoting by B the buffer size, the volume available for each TCP flow is given by:

$$\frac{B-x}{n}.$$

The video stream has priority as long as it has a lower data volume in the buffer than each TCP flow, that is if:

$$x < \frac{B}{n+1}. \tag{1}$$

Let C be the link capacity in packet/s. If the video stream were purely CBR (constant bit rate) at rate $\lambda < C$, there would be at most one packet in the buffer and the video stream would always be prioritized. More generally, if the video stream is VBR but has a strictly enforced peak rate $\lambda < C$, there is at most one packet in the buffer and the video stream is always prioritized. Now assume that the peak rate λ is not strictly enforced. Specifically, we assume that packets arrive according to a Poisson process of intensity λ. Thus, while the *theoretical* peak rate is equal to λ, the *actual* peak rate is infinite (there is no lower bound on the packet inter-arrival time). Note that this traffic model with transmission at the theoretical peak rate λ is in fact much more pessimistic than any more realistic, long-range dependent traffic model that accounts for the rate fluctuations between 0 and λ due to the structure of the video [5].

We seek to estimate the probability $p(T)$ that the video is de-prioritized, in the sense that condition (1) is violated, over some time period of length T. In view of the above assumptions, this reduces to the analysis of the transient behavior of an $M/D/1$ queue. In order to get simple and explicit results, we first calculate the probability $q(t)$ that the number of arrivals exceeds the service capacity plus the buffering capacity (1) over any time interval of length t:

$$q(t) = \Pr\left(N(t) \geq Ct + \frac{B}{n+1}\right)$$

where $N(t)$ denotes a Poisson random variable of mean λt. We then divide the time period T into k intervals of length T/k and calculate the probability q_k that the number of arrivals exceeds the service capacity plus the buffering capacity (1) on at least one of these intervals:

$$q_k = 1 - \left(1 - q\left(\frac{T}{k}\right)\right)^k.$$

The probability of de-prioritization $p(T)$ is then estimated as the maximum of the probabilities q_k over all k:

$$p(T) = \max_{k \geq 1} q_k. \tag{2}$$

Note that the maximum is reached for some finite k since q_k tends to 0 as k grows to infinity.

Fig.3 gives the probability of de-prioritization as a function of the video peak rate λ for a 10-min video on a 10Mbit/s link with different numbers n of competing TCP flows. The buffer size is equal to 200 packets, with a packet size of 1250B. For $n = 20$ competing flows for instance, the probability of de-prioritization is close to 0 as long as the video peak rate is less than 6Mbit/s. The figure also shows the probability of de-prioritization under FQ, which is equal to 0 if $\lambda < C/(n+1)$ (since the video peak rate is less than the fair share) and to 1 otherwise. For $n = 20$ competing flows for instance, the video peak rate cannot exceed 480kbit/s.

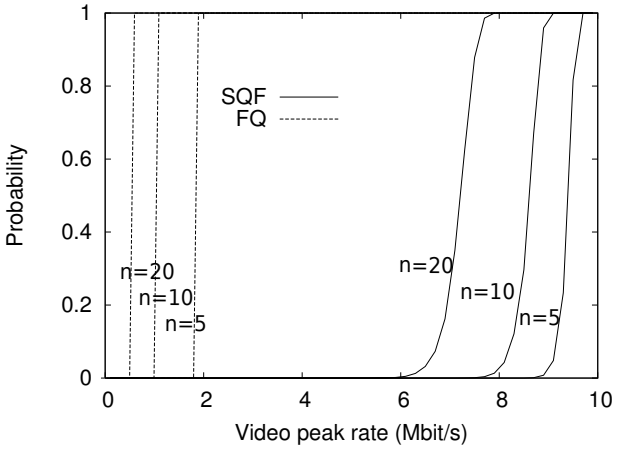


Fig. 3. Probability of de-prioritization for a 10 min video on a 10Mbit/s link with $n = 5, 10, 20$ competing TCP flows (from right to left).

Another way of evaluating the performance of SQF is to estimate the average time $\bar{T}$ during which the video is prioritized. Note that the probability that the time of prioritization is larger than t is equal to $1 - p(t)$, where $p(t)$ is given by (2). We deduce:

$$\bar{T} = \int_0^\infty (1 - p(t))dt.$$

Fig.4 gives the average time of prioritization $\bar{T}$ as a function of the video peak rate λ on a 10Mbit/s link with different numbers n of competing TCP flows. For $n = 20$ competing flows for instance, this average time is larger than 180min as long as the video peak rate is less than 6Mbit/s. Under FQ, the video peak rate cannot exceed 480kbit/s in this case.

B. Multiple video streams

The results readily extend to an arbitrary number m of video streams. Let x_i be the number of packets of video stream i in the buffer. Let $x = \sum_{i=1}^m x_i$. All video streams have priority as long as each has a lower data volume in the buffer than each TCP flow, that is if:

$$\forall i, \quad x_i < \frac{B-x}{n}. \tag{3}$$

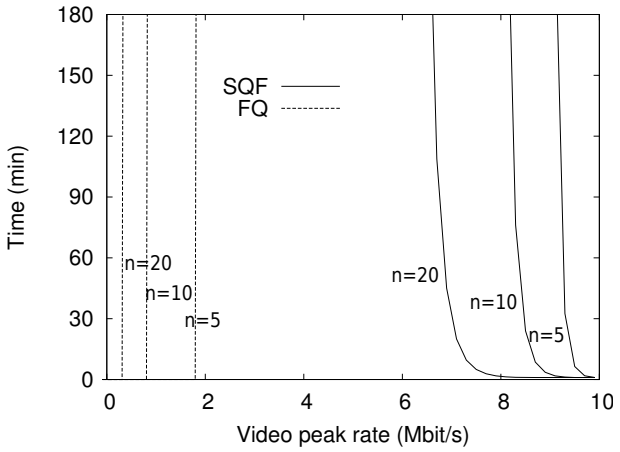


Fig. 4. Average time of prioritization of a video on a 10Mbit/s link with $n = 5, 10, 20$ competing TCP flows (from right to left).

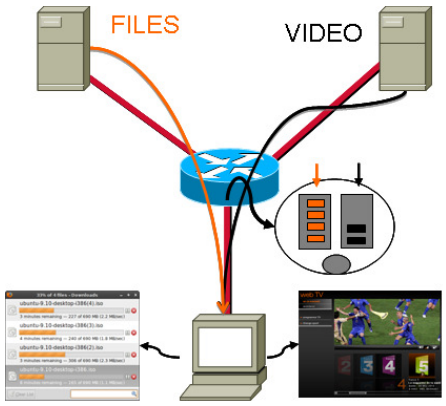


Fig. 5. Testbed set-up made of two servers and one client connected in a star topology.

Clearly, the condition (1) imposed on the total number of video packets is sufficient. This is equivalent to consider the m video streams of rate λ as a single video stream of rate $m\lambda$. Thus for a 10Mbit/s link with $n = 20$ competing TCP flows, SQF gives priority to $m = 3$ video streams of peak rate 2Mbit/s (since the total video stream does not exceed 6 Mbit/s).

V. Experimental results

This section presents some experimental results on the performance of FIFO, FQ and SQF. We present here a limited number of tests for lack of space and for sake of clarity. The set of test has been chosen in order to better expose the behavior and performance of the schedulers in presence of more realistic traffic patterns.

A. Test-bed setup

The test-bed consists of four Linux PCs, that emulate a router, two servers and a client, as depicted by Fig.5. One server is an Apache Web server [2], the other one is a video streaming server using VLC [3]. The Linux router is equipped with three network Ethernet cards so as to connect the two servers and the client in a star topology, using standard Linux tools [1]. All line cards are used at 100Mbit/s except for the link of the client that is used at 10Mbit/s and plays the role of the user's access line. The corresponding output line card of the router has a buffer size of 50 MTU sized packets (with a MTU of 1500B) and embeds one egress queuing discipline, chosen among FIFO (without buffer management), FQ (DRR algorithm with LQD), and SQF (as described in Sec. III). Flows are identified through the 5-tuple of the IP header.

B. Traffic scenario

The client downloads several files from the Web server while watching a video channel streamed on RTP over UDP by the VLC server. The video is composed of two flows, one for the video and one for the audio, on different UDP ports. The codec is MPEG2 for both video and audio. The video stream is a VBR flow with an average rate of 5Mbit/s while the audio stream is a VBR flow with an average rate of 64kbit/s.

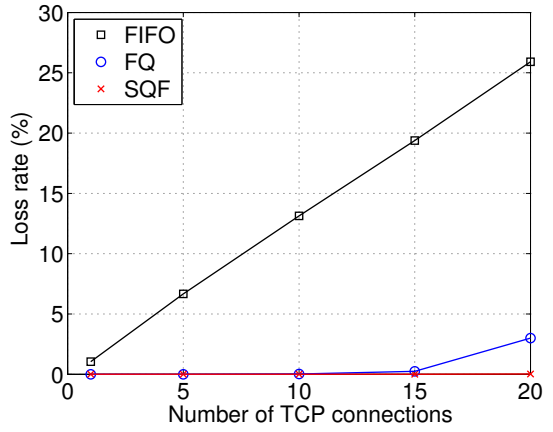
C. Quality of the video stream

The quality of the video stream is estimated through the loss rate and jitter experienced by packets of the audio and video flows on a 10-min sequence. These performance metrics are calculated from tcpdump records [4], available at the client input interface. The loss rate is deduced from the stream sequence numbers while jitter is estimated through the inter-packet delay variation, as defined in [10].

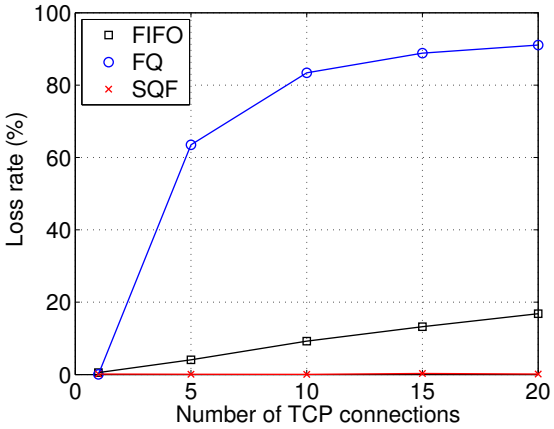
Fig.6 shows the packet loss rate of the audio and video flows with respect to the number n of competing file downloads, varying from 1 to 20. We observe that the loss rate is negligible under SQF independently of n, as expected in view of the analysis of Sec. IV, whereas it is much too high under FIFO and FQ as soon as $n > 1$. Under FIFO, the packet loss rate of the audio flow grows linearly from 1% to more than 25% as n grows from 1 to 20; it is slightly less for the video flow. Video traffic having a higher rate than audio forces TCP to back-off and reduce data sending rate.

Under FQ, the behaviors of the audio and video flows differ significantly: the audio flow has an average rate of 64kbit/s that is less than the fair share, which explains the resulting low loss rate; the video flow, on the other hand, has an average rate of 5Mbit/s that is larger than the fair share as soon as $n > 1$, which explains the resulting high loss rate. These results confirm the poor performance of FQ under high data traffic load.

The results are in agreement with those obtained for the jitter, reported in Fig.7. We observe that under FQ, the jitter increases with the number of flows for both the audio flow and the video flow, due to the round robin scheduling time. PFQ would limit the jitter suffered by the audio flow but not that of the video flow, whose average rate is larger than the fair share as soon as $n > 1$. SQF turns out to be beneficial for both audio and video, independently of the number of competing file downloads.

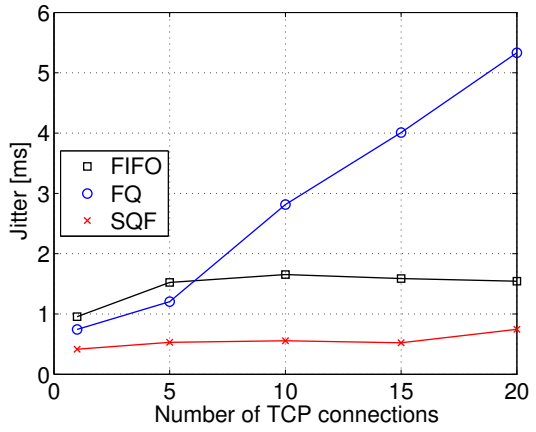


(a) Audio

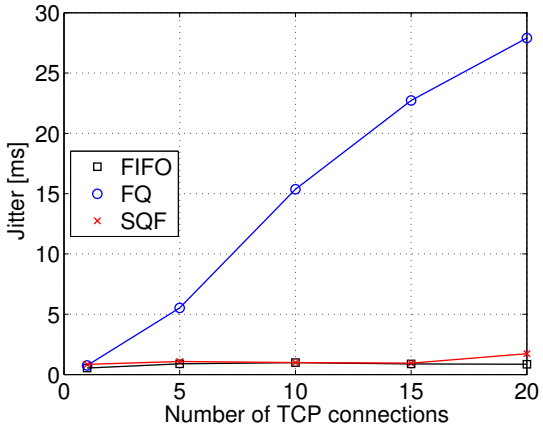


(b) Video

Fig. 6. Packet loss rate of the audio and video flows with respect to data traffic load.



(a) Audio



(b) Video

Fig. 7. Jitter of the audio and video flows with respect to data traffic load.

D. Data traffic

In Tab.I we have reported some performance measures for data traffic. The data bandwidth share corresponds to the total throughput of data traffic while Jain's index indicates how this bandwidth is distributed among the n data transfers. Specifically, Jain's index is defined by:

$$J = \frac{(\sum_{i=1}^{n} y_i)^2}{n \sum_{i=1}^{n} y_i^2}$$

where y_i denotes the normalized throughput achieved by the i-th data download at the end of the experiment. Note that J varies from $1/n$ for a totally unfair sharing (a single flow gets all the resource) to 1 for a perfectly fair sharing (all throughputs are equal).

As expected, the data bandwidth share increases with n under both FIFO and FQ whereas it remains constant under SQF, due to the self-prioritization of the video stream. Under FQ, the bandwidth available for the video stream is not sufficient as soon as $n > 1$, which is in agreement with the results of Sec. V-C. Note that in all cases, the overall utilization does not reach 100% due to the transmission of duplicate packets by TCP. Under SQF for instance, the data bandwidth share is equal to 4.1Mbit/s so that the overall utilization, including the 5Mbit/s video flow and the 64kbit/s audio flow, is approximately equal to 92%. Finally, fairness as measured by Jain's index is excellent for all schedulers.

TABLE I
DATA BANDWIDTH SHARE (IN MBIT/S) AND JAIN'S INDEX FAIRNESS AS A FUNCTION OF THE NUMBER n OF DATA DOWNLOADS.

n	1	5	10	15	20
FIFO	4.1/0.95	4.4/0.91	4.8/0.95	5.1/0.95	5.6/0.99
FQ	4.1/0.99	7.5/0.99	8.5/0.95	8.9/0.99	9.1/0.99
SQF	4.1/0.95	4.1/0.90	4.1/0.91	4.1/0.95	4.1/0.95

E. Impact of video rate variability

We have so far assumed that the video stream does not suffer any variable delay on its path from the video server to the considered bottleneck link. In order to illustrate the impact of jitter on SQF efficiency, we add artificial packet delays at the video source. Specifically, packets are delayed according to a Gaussian random variable with different values of the standard deviation δ, so that packets may be sent in bursts similarly to those generated by TCP.

We report in Fig.8 the loss rate experienced by the video stream with respect to the standard deviation δ, for a link capacity of 10Mbit/s and 100Mbit/s under SQF. When δ increases, packets are more likely to be sent in bursts and to accumulate in the buffer, causing the de-prioritization of the video stream. For a 10Mbit/s link for instance, the video stream is de-prioritized when $\delta \geq 5$ms; this threshold value is larger for a 100Mbit/s link.

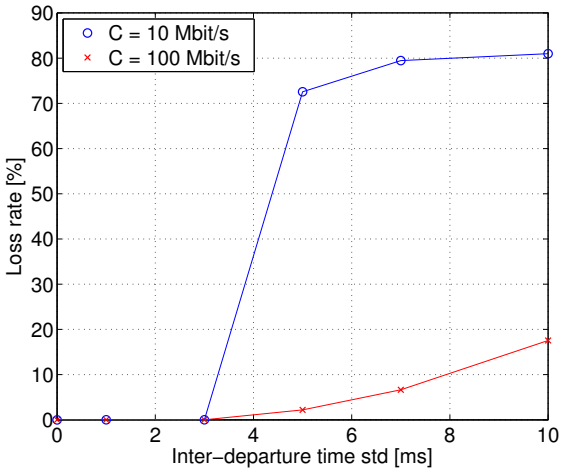


Fig. 8. Packet loss rate of the video flow with respect to the packet inter-departure time standard deviation.

VI. Conclusion

We have presented a packet scheduler that automatically prioritizes audio and video traffic for interactive applications. The algorithm need not any prior information about the services to prioritize unlike the Diffserv architecture. It simply relies on the following key difference between streaming and data traffic: while the former generates no or little queuing as long as the stream rate, controlled by the codec, is less than the bottleneck capacity, the latter tends to fully exploit the available bandwidth and thus to saturate the buffer of the bottleneck link. This difference is easily detected by the SQF algorithm. By contrast, the FQ scheduling policy is based on the throughput of each flow and thus is unable to prioritize a video stream in the presence of high data traffic load.

We claim that FQ should rather be used to fairly share resources among multiple *users*, in back-haul or passive optical networks for instance. Note the high bandwidth of the corresponding links could well be saturated. The advent of optical Internet access will likely drive a different domestic usage with richer multimedia content, a significant part of this content being produced by the users themselves and stored remotely into virtualized servers. Both the uplink and the downlink will then be shared by a number of applications, both streaming and elastic, whose multiplexing will be critical. SQF could then be used to self-prioritize delay-sensitive services within each user's bandwidth share. Note, again, that the scalability of the algorithm is guaranteed by the limit on the number of active flows imposed by the buffer size. Moreover, this number has been proved to be typically limited by the nature of traffic itself, independently of the link capacity, see [13], [14], [15]. For future work, we plan to assess the efficiency and robustness of the envisaged hierarchical scheduling policy consisting of FQ at user level and SQF at flow level.

Acknowledgment

This work has been funded by the EU FP7 4WARD project on the Future Internet.

References

[1] Linux advanced routing and traffic control http://lartc.org
[2] The HTTP Apache project http://httpd.apache.org/
[3] The VideoLAN Project
http://www.videolan.org/
[4] Tcpdump/Libpcap
http://www.tcpdump.org/
[5] J. Beran, R. Sherman, M.S. Taqqu, W. Willinger. Long-range dependence in variable-bit-rate video traffic. IEEE Transactions on Comm. Vol. 43. 1995.
[6] S. Blake et. al. Architecture for Differentiated Service, RFC 2475 (Informational), updated by RFC 3260, IETF, Dec. 1998.
[7] T. Bonald and L. Muscariello. Shortest queue first: implicit service differentiation through per-flow scheduling. Demo at IEEE LCN 2009.
[8] G. Carofiglio and L. Muscariello. On the impact of TCP and per-flow scheduling on Internet performance. In Proc. of IEEE INFOCOM 2010.
[9] A. Demers , S. Keshav and S. Shenker. Analysis and simulation of a fair queueing algorithm. In Proc. of ACM SIGCOMM 1989.
[10] C. Demichelis, P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), RFC 3393, IETF, Nov. 2002.
[11] A. Dimakis and J. Walrand. Sufficient conditions for stability of longest-queue-first scheduling: second- order properties using fluid limits. Annals of Applied Probablity, 2006.
[12] R. Jain, D.M. Chiu, W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems. DEC Research Report TR-301, 1984.
[13] A. Kortebi, L. Muscariello S.Oueslati and J. Roberts. On the Scalability of Fair Queueing. In Proc. of ACM SIGCOMM HotNets III, 2004.
[14] A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In Proc. of ACM SIGMETRICS 2005.
[15] A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts. Minimizing the overhead in implementing flow-aware networking. In Proc. of IEEE/ACM ANCS 2005.
[16] A. Kortebi, S. Oueslati and J. Roberts. Cross-protect: implicit service differentiation and admission control. In Proc. of IEEE HPSR 2004.
[17] N. McKeown, V. Anantharam and J. Walrand. Achieving 100% throughput in an input-queued switch. In Proc. of IEEE INFOCOM 1996.
[18] J. Nagle. On Packet Switches with Infinite Storage. IEEE Transactions on Communications, Vol 35, 1987.
[19] R. Pan, L. Breslau, B. Prabhakar and S. Shenker. Approximate fairness through differential dropping. ACM SIGCOMM CCR 2003.
[20] R. Pan, B. Prabhakar, K. Psounis. CHOKe: A stateless queue management scheme for approximating fair bandwdith allocation. Proceedings of IEEE INFOCOM 2000.
[21] G. Raina, D. Towsley and D. Wischik Part II: Control theory for buffer sizing. ACM SIGCOMM CCR 2005.
[22] G. Sharma, R. Mazumdar, and N. Shroff. Joint Congestion Control and Distributed Scheduling for Throughput Guarantees inWireless Networks. IEEE INFOCOM 2007.
[23] M. Shreedhar and G. Varghese Efficient fair queueing using deficit round robin. In Proc. of ACM SIGCOMM 1995.
[24] B. Suter, T.V. Lakshman, D. Stiliadis, A.K. and Choudhury. Design considerations for supporting TCP with per-flow queueing. In Proc. of IEEE INFOCOM 1998.
[25] L. Tassiulas and A. Ephremides, Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. IEEE Transactions on Automatic Control, Vol. 37, No. 12, 1992.