Priority Switching Scheduler

Anaïs Finzi, Victor Perrier, Fabrice Frances, Emmanuel Lochin

Priority Switching Scheduler

Anaïs Finzi, Victor Perrier, Fabrice Francès, Emmanuel Lochin

Abstract

We define a novel core network router scheduling architecture called Priority Switching Scheduler (PSS), to carry and isolate time constrained and elastic traffic flows from best-effort traffic. To date, one possible solution has been to implement a core DiffServ network with standard fair queuing and scheduling mechanisms as proposed in the well-known "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic" from RFC5865. This architecture is one of the most selected solutions by internet service provider for access networks (e.g., Customer-Premises Equipment) and deployed within several performace-enhancing proxies (PEP) over satellite communications (SATCOM) architectures. In this study, we argue that the proposed standard implementation does not allow to efficiently quantify the reserved capacity for the AF class. By using a novel credit-based shaper mechanism called Burst Limiting Shaper (BLS) to manage the AF class, we show that PSS can provide the same isolation for the time constrained EF class while better quantifying the part allocated to the AF class. PSS operates both when the output link capacity is fixed (e.g., wire links, terrestrial networks) or might vary due to system impairments or weather condition (e.g., wireless or satellite links). We demonstrate the capability of PSS through an emulated SATCOM scenario with variable capacity, and show the AF output rate is less dependent on the EF traffic, which improves the quantification of the reserved capacity of AF, without impacting EF traffic.

1 Introduction

To improve the usage of the scarce satellite resource and propose a good user service, implementing an adequate Quality of Service (QoS) mechanism is an important part of SATCOM systems [20, 23, 9, 10]. As a matter of fact, satellite bandwidth management has an important economical aspect.

QoS in IP networks relates to the ability to manage and prioritize different types of network traffic to provide a consistent level of service to different types of applications or users. In satellite communications (SATCOM), performance-enhancing proxies (PEP) are devices or software components that can be used to improve the performance of the SATCOM system by reducing the amount of traffic that needs to be sent over the satellite link. PEPs can be used to optimize network traffic in a variety of ways, such as compressing data to reduce the amount of bandwidth that is required, caching frequently requested data to reduce the number of requests that need to be sent over the satellite link, and reducing the number of round trips required to complete a transaction [5]. PEPs can also be used to manage and prioritize network traffic to ensure that important or time-sensitive traffic is given priority over less important traffic. When QoS is integrated with PEPs in SATCOM systems, it allows managing the network traffic that is sent to and received from the satellite link, by prioritizing different types of traffic, shaping the traffic to ensure that the network resources are used efficiently, and also by reducing the amount of traffic that needs to be sent over the satellite link to improve the performance of the SATCOM system. In summary, PEPs and QoS can work together to improve the performance and reliability of SATCOM systems by managing and prioritizing network traffic and reducing the amount of traffic that needs to be sent over the satellite link.

QoS in IP networks can be integrated into SATCOM systems using a combination of hardware and software components [1]. One common approach is to use a QoS Policy to manage and prioritize network traffic. The PEP can be implemented as a separate device or as a software component [8] that runs on existing network equipment. It is responsible for enforcing QoS policies on the traffic by classifying, marking, and shaping the traffic according to the policies. Another approach is to use QoS routers, which are specially designed to manage and prioritize network traffic. These routers can be integrated into the SATCOM system by placing them at key points in the network, such as at the edge of the network where traffic enters and exits the satellite link [5, 11]. The QoS routers can be configured to use various techniques, such as prioritizing certain types of traffic, rate limiting, or shaping the traffic to ensure that the network resources are used efficiently.

Additionally, SATCOM providers usually offer specific QoS plans that include options to manage traffic, such as bandwidth allocation or Service-Level Agreements (SLA) which also can be used to ensure that different types of applications or users receive the appropriate level of service [1, 11]. In summary, integrating QoS in SATCOM systems usually involves using a combination of hardware and software components [1], such as PEPs and QoS routers, and service plans offered by the providers, to manage and prioritize the network traffic that is sent to and received from the satellite. The current contribution takes place in this context.

The typical layer three QoS approach adopted by SATCOM systems is based on DiffServ architecture and usually follows RFC5865 specification [3]. In particular, the QoS scheme described in this RFC recommends managing AF (for elastic TCP traffic) and CS/0 (all non QoS guaranteed traffic) classes with a rate scheduler scheme. Basically, such a DiffServ stateless core network is used to differentiate time-constrained UDP traffic (e.g., VoIP or VoD) and TCP bulk data transfer from all the remaining best-effort (BE) traffic called default traffic (CS/0).

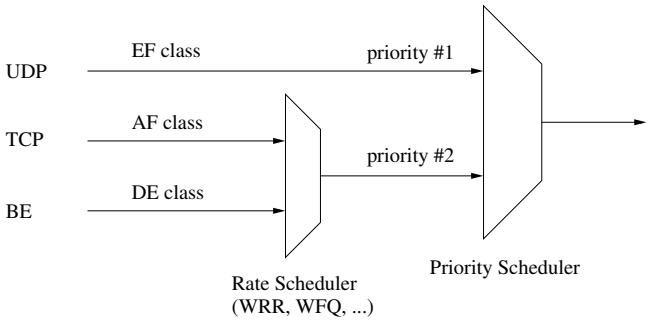


Figure 1: DiffServ core router architecture following [3].

This core router implementation, illustrated in Fig. 1, is widely implemented inside customer-premises equipment (CPE) [7] or satellite performance-enhancing proxy (PEP) [6, 16] to differentiate flows with different QoS requirements. It provides the first and best service to EF as the priority scheduler attributes the highest priority to this class. The second service is called assured service and is built on top of the AF class where elastic traffic, such as TCP traffic, is intended to achieve a minimum level of throughput [18]. Usually, the minimum assured throughput is given according to a negotiated profile with the client. The throughput increases as long as there are available resources and decreases when congestion occurs. As a matter of fact, a simple priority scheduler is insufficient to implement the AF service. Due to its opportunistic nature of fetching the full remaining capacity, TCP traffic increases until reaching the capacity of the bottleneck. In particular, this behavior could lead to starve the CS0 class. To prevent this, the core router architecture proposed in [3] uses a rate scheduler between AF and CS0 classes to share the residual capacity left by the EF class. Nevertheless, one drawback of using a rate scheduler is the high impact of EF traffic on AF. Indeed, the residual capacity shared by AF and CS0 classes is directly impacted by the EF traffic variation. As a consequence, the AF class service is difficult to predict in terms of available capacity and latency. Furthermore, currently implemented rate scheduler schemes, such as Weighted Round Robin (WRR) [15], Deficit Weighted Round Robin (DWRR), Deficit Round Robin (DRR) [17], induce excessive margin to ensure the capacity requested, which reduces the number of users that could be accepted in the network.

To overcome these limitations and considering the gap in terms of QoS guarantee offered by previously cited rate scheduling schemes, we propose in this study to assess the benefit of a credit-based scheduler mechanism [4], called Burst Limiting Shaper (BLS) and presented by the Time Sensitive Networking (TSN) Task group [13], to manage the AF class. The use of a Burst Limiting Shaper allows reserving a given capacity for the shaped priority. As with a rate scheduler, this reservation sets the capacity allocated to the shaped priority in the presence of CS0 traffic. However, contrary to the rate scheduler, the BLS can enforce the reserved capacity when the EF traffic dynamically evolves over the time.

Hence, this paper uses the BLS principle but adapted and modified for the AF class. The resulting algorithm is called Priority Switching Scheduler. PSS provides better guarantees in terms of quantifiable rates, while avoiding any negative impact on the performances of the EF class, in comparison to a rate scheduler, e.g., Weighted Round Robin (WRR) or Weighted Fair Queuing (WFQ). The benefit of using PSS for satellite communications results in a more predictable output rate per traffic class and thus a better management of the available, and possibly variable, satellite capacity in accordance with end-users quality of experience (QoE).

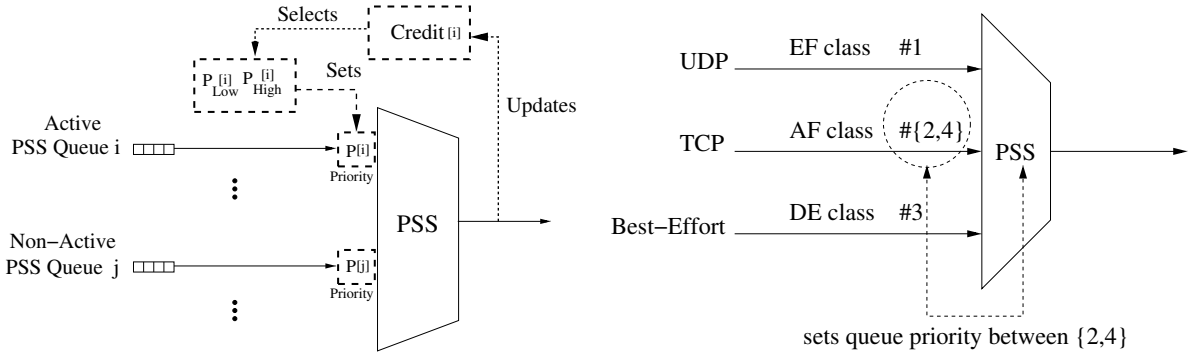
To tackle this problem, we first give the big picture of our idea by presenting our PSS router proposal. Then, we detail the PSS principle and present experiments and results. In particular, we show that with a given EF traffic, a correspondence between the PSS parameters and the weights of a WRR can be simply established. Finally, we present the new service offered by the PSS when the EF traffic varies and compare it to the WRR service.

PSS was initially proposed by the same authors in [12], but the algorithm was limited to fixed link capacity and evaluated within the old and well-known ns-2 network simulator [2]. This paper greatly extends this preliminary contribution by proposing an improved algorithm considering variable link capacity, a practical implementation of PSS freely available to reproduce our experiments[1] and a performance analysis with real measurements supported by theory.

[1] `https://github.com/elochin/priority-switching-scheduler`

2 Priority Switching Scheduler

Before diving into PSS internal starting at Section 2.1, and to ease the understanding of this scheduler, we present below the basic principle of PSS and the implementation that will be used in the tests.



(a) PSS generic view

(b) PSS proposed implementation in this paper

Figure 2: Architecture of PSS

Figure 2(a) illustrates the principle of PSS. The algorithm is based on the use of credit counters (detailed in the following) that changes the priority of one or several queues. As previously said, the idea follows a proposal made by the TSN Task group named Burst Limiting Shaper (BLS) [4], a credit-based shaper presented in [13]. As for the BLS, PSS defines by an upper threshold: L_M, a lower threshold: L_R such as $0 \leqslant L_R < L_M$, and a reserved capacity: BW. These values are explained in the next subsection 2.1.

As shown Figure 2(a), for each controlled queue i, each priority denoted $p[i]$ changes between two values denoted $p_{low}[i]$ and $p_{high}[i]$, depending on the associated credit counter, i.e., $credit[i]$. Then, a Priority Scheduler is used for the dequeuing process, e.g., among the queues with available traffic, the first packet of the queue with the highest priority is dequeued. The main idea is that changing the priorities adds fairness to the Priority Scheduler. Depending on the credit counter parameters, the amount of capacity available to a controlled queue is bounded between a minimum and a maximum value. Consequently, good parameters setting is crucial to prevent starvation of lower priority queues.

The service obtained for the controlled queue with the switching priority is more predictable and corresponds to the minimum between a target capacity and the residual capacity left by higher priorities. The impact of the input traffic sporadicity from higher classes is thus transferred to non-active PSS queues with a lower priority. Finally, PSS offers much flexibility, as both (1) controlled queues with a guaranteed capacity (when two priorities are set) and (2) queues scheduled with a simple Priority Scheduler (when only one priority is set) can conjointly be enabled.

Considering the first previous Fig. 1 where three classes of service are presented, the resulting PSS architecture will be implemented as illustrated Fig. 2(b).

The notations used in this paper are presented in Table 1.

Table 1: Notations

C	capacity of a link
L_M	PSS maximum credit level
L_R	PSS resume credit level, set to zero when non-varying link capacity
I_{idle}	PSS idle slope
I_{send}	PSS sending slope
BW	PSS reserved capacity
L_j^{max}	maximum length of a packet of class $j \in \{EF, AF, CS0\}$
L_j^{avg}	average length of a packet of class $j \in \{EF, AF, CS0\}$
W_i	WRR weight of class $i \in \{AF, CS0\}$
K_i	relative weight of class $i \in \{AF, CS0\}$, defined in (2)
R_j	input rate of class $j \in \{EF, AF, CS0\}$
R_j^{exp}	expected input rate of class $j \in \{EF, AF, CS0\}$
$R_{k/j}^{*,th}$	theoretical output rate of class $j \in \{EF, AF, CS0\}$ using $k \in \{WRR, PSS\}$
$R_{k/j}^{*,sim}$	simulated output rate of class $j \in \{EF, AF, CS0\}$ using $k \in \{WRR, PSS\}$

2.1 Specification

The PSS algorithm defines for each queue q a low priority, $p_{low}[q]$, and a high priority, $p_{high}[q]$. Each PSS controlled queue q with $p_{high}[q] < p_{low}[q]$ is associated to a credit counter $credit[q]$ which manages the priority switching. Each credit counter is defined by:

- a minimum level: 0;
- a maximum level: $LMs[q]$;
- a resume level: $LRs[q]$;
- a reserved capacity: $BWs[q]$;
- an idle slope: $I_{idle}[q] = C \cdot BWs[q]$;
- a sending slope: $I_{send}[q] = C - I_{idle}[q]$;

The available capacity is mostly impacted by the guaranteed capacity $BWs[q]$. Hence, $BWs[q]$ should be set to the targeted capacity plus a margin considering the additional packet due to non-preemption as explained in the following: the value of $LMs[q]$ can negatively impact on the guaranteed available capacity. The maximum level determines the size of the maximum sending windows, i.e, the maximum uninterrupted transmission time of the controlled queue packets before a priority switching. The impact of the non-preemption is as a function of the value of $LMs[q]$. The smaller the $LMs[q]$, the larger the impact of the non-preemption is. For example, if the number of packets varies between 4 and 5, the variation of the output traffic is around 25% (i.e., going from 4 to 5 corresponds to a 25% increase). If the number of packets sent varies between 50 and 51, the variation of the output traffic is around 2%.

The credit allows keeping track of the packet transmissions. However, in two cases, the keeping track of the transmission is problematic: when the credit is saturated at $LMs[q]$ or at 0. In both cases, packets are transmitted without gained or consumed credit. Nevertheless, the resume level can be used to decrease the times when the credit is saturated at 0. If the resume level is 0, then as soon as the credit reaches 0, the priority is switched and the credit saturates at 0 due to the non-preemption of the current packet. On the contrary, if $LRs[q] > 0$, then during the transmission of the non-preempted packet, the credit keeps on decreasing before reaching 0 as illustrated in Figure 3

Hence, the proposed value for $LRs[q]$ is

$$LRs[q] = L_{max}(MC(q)) \cdot BWs[q],$$

with $MC(q)$ the queues such as

$$k \text{ in } MC(q) \rightarrow p_{low}[q] > (p_{low}[k] \text{ OR } p_{high}[k]) > p_{high}[q],$$

and $L_{max}(MC(q))$ the maximum packet size among all the controlled queues. With this value, there is no credit saturation at 0 due to non-preemption.

Finally, we propose to use the following parameters of a controlled queue q:

- $LRs[q] = L_{max}(MC(q)) \cdot BWs[q]$.
- $BWs[q] = BW_{target}[q] + 1/(N-1)$;
- $LMs[q] = (N-1) \cdot L_{max}(q) \cdot (1 - BWs[q]) + LRs[q]$;

with N the maximum number of packet of queue q set uninterrupted (considering the non-preemption) and $BW_{target}[q]$ the percentage of targeted available capacity.

A similar parameter setting is described in [12], to transform WRR parameter into PSS parameters, in the specific case of 3-classes DiffServ architecture.

The priority change depends on the credit counter as follows:

- initially, the credit counter starts at 0;
- the change of priority $p[q]$ of queue q occurs in two cases:
  1. if $p[q] = p_{high}[q]$ and the credit reaches $LMs[q]$;
  2. if $p[q] = p_{low}[q]$ and credit reaches $LRs[q]$;

- when a packet of queue q is transmitted, the credit increases with a rate $I_{send}[q]$, else the credit decreases with a rate $I_{idle}[q]$;
- when the credit reaches $LMs[q]$, it remains at this level until the end of the transmission of the current packet;
- when the credit reaches 0, it remains at this level until the start of the transmission of a queue q packet.

Figure 3 and Figure 4 show two examples of credit and priority changes of a given queue q. First, in Figure 3, we show an example when the controlled queue q sends its traffic continuously until the priority change. Then other traffic is also sent uninterruptedly until the priority changes back. In Figure 4, we propose a more complex behavior. First, this figure shows when a packet with a priority higher than $p_{high}[q]$ is available, this packet is sent before the traffic of class q. Secondly, when no traffic with a priority lower than $p_{low}[q]$ is available, then traffic of queue q can be sent. This highlight the non-blocking nature of PSS and that $p[q] = p_{high}[q]$ (resp. $p[q] = p_{low}[q]$) does not necessarily mean that traffic of queue q is being sent (resp. not being sent).

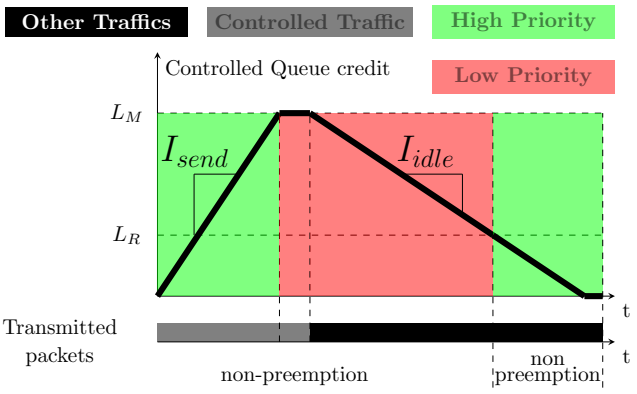


Figure 3: First example of queue q credit and priority behaviors

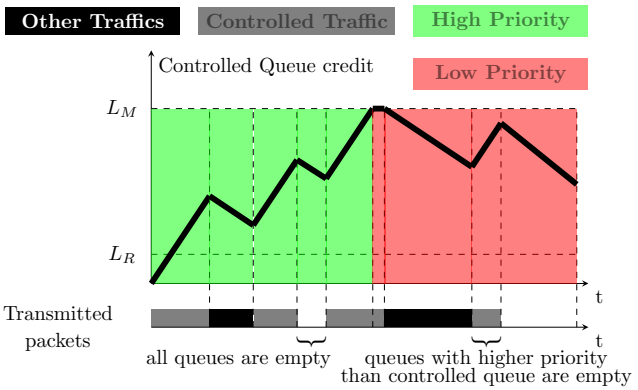


Figure 4: Second example of queue q credit and priority behaviors

Finally, for the dequeuing process, a Priority Scheduler selects the appropriate packet using the current p[q] values, e.g., among the queues with available traffic, the first packet of the queue with the highest priority is dequeued.

2.2 Implementation

The dequeuing algorithm is presented in the PSS Algorithm 1. This algorithm works as follows :

- the credit of each queue q, denoted $credit[q]$, and the dequeuing timer denoted $timerDQ[q]$ are initialized to zero;
- the initial priority is set to the high value $p_{high}[q]$;
- first, for each queue with $p_{high}[q] > p_{low}[q]$, the difference between the current time and the time stored in $timerDQ[q]$, is computed (lines 2 and 3);

- the duration $dtime[q]$ represents the time elapsed since the last credit update, during which no packet of the controlled queue q was sent, we call this the idle time. Then, if $dtime[q] > 0$, the credit is updated by removing the credit gained during the idle time that just occurred (lines 5 and 9). We will come back in Section 2.3 when condition line 4 is false;
- next, $timerDQ[q]$ is set to the current time to keep track of the time the credit is last updated (line 10);
- if the credit reaches $LRs[q]$, the priority changes to its high value (line 11);
- then, with the updated priorities, the priority scheduler performs as usual: each queue is checked for dequeuing, the highest priority first (lines 19 and 20);
- when a queue q is selected with $p_{high}[q] < p_{low}[q]$, the credit expected to be consumed is added to $credit[q]$ variable (line 22);
- the time taken for the packet to be dequeued is added to the variable $timerDQ[q]$ (lines 23) so the transmission time of the packet will not be considered in the idle time $dtime[q]$ (line 2);
- if the credit reaches $LMs[q]$, the priority changes to its low value (line 25). Finally, the packet is dequeued (line 28).

Algorithm 1 PSS algorithm: dequeuing process

Require: credits, timerDQs, C, LMs, LRs, BWs, p_{highs}, p_{lows}
  for each queue q with $p_{high}[q] < p_{low}[q]$ **do**
    $dtime[q] = currentTime - timerDQ[q]$
    if $dtime[q] > 0$ **then**
      if $length(q) > 0$ **then**
        $credit[q] = max(credit[q] - dtime[q] \cdot C \cdot BWs[q], 0)$
      else
        $credit[q] = max(credit[q] - dtime[q] \cdot C \cdot BWs[q], min(credit[q], LRs[q]))$
      end if
      $dtime[q] = currentTime$
      if $credit[q] < LRs[q]$ and $p[q] = p_{low}[q]$ **then**
        $p[q] = p_{high}[q]$
      end if
    else
      $credit[q] = min(credit[q] - dtime[q] \cdot C \cdot BWs[q], LMs[q])$
      $dtime[q] = currentTime$
    end if
  end for
  for each priority level pl, highest first **do**
    if $length(queue(pl)) > 0$ **then**
      $q = queue(pl)$
      if $p_{high}[q] < p_{low}[q]$ **then**
        $credit[q] = min(LMs[q], credit[q] + size(head(q)) \cdot (1 - BWs[q]))$
        $timerDQ[q] = currentTime + size(head(q))/C$
        if $credit >= LM[q] and p[q] = p_{high}[q]$ **then**
          $p[q] = p_{low}[q]$
        end if
      end if
      $dequeue(head(q))$
      break
    end if
  end for

The complexity of this algorithm is the same as a priority scheduler and is $O(1)$ (the number of queues is constant).

PSS algorithm also implements the following functions:

- $getCurrentTime()$ uses a timer to return the current time;
- $queue(pl)$ returns the queue associated to priority pl;

- $head(q)$ returns the first packet of queue q;
- $size(f)$ returns the size of packet f;
- $dequeue(f)$ activates the dequeuing event of packet f.

2.3 Modifications introduced to handle variable link capacity

The satellite link capacity may vary due to impairments or weather condition. This is also the case for cellular or other wireless links. However, the link capacity C is a strong hypothesis and have an impact on the performance of PSS. As we seek to use PSS within SATCOM PEP, we need to evaluate the impact of this variation and how PSS should consider this network characteristic to enable the deployment of such solution for SATCOM. So in the following, we assume that the capacity of the satellite link is varying over the time and consider only a mean capacity with a possible standard deviation. We also consider that EF class traffic may also be varying, as this could be seen by PSS as a variation of the available capacity. To adapt PSS to these characteristics, we propose a few changes to the burst-limiting shaper algorithm. We explain and justify in this section the modifications proposed.

As we are now considering both a variable capacity and variable traffic in the EF class, the available flow rate for a given queue q may be inferior to $BWs[q] \cdot C$. While this happens, the credit stalls to 0. Then, if the available capacity moves above the target $BWs[q] \cdot C$, PSS will allow the flow to transmit to its target flow rate, but never more. As a matter of fact, this does not allow compensating the previous starvation state.

To balance that phenomenon, the scheduler should prioritize q longer than usual. Instead of having:

$$LMs[q] = (N-1) \cdot L_{max}(q) \cdot (1 - BWs[q])$$

and

$$LRs[q] = L_{max}(MC(q)) \cdot BWs[q],$$

the algorithm memorizes the deficit of data occurring to the controlled queue q, allowing to compensate this gap during the next transmission round and better achieved the target throughput BW on the long term.

We then have:

$$LMs[q] = (N-1) \cdot L_{max}(q) \cdot (1 - BWs[q]) + LRs[q]$$

with $LRs[q]$ still to define: the scale of $LRs[q]$ must be defined by the policy of the scheduler, and must consider the shape of the capacity of the link, and the shape of all flows with a higher priority level.

Of course, $LRs[q]$ should still be above $L_{max}(MC(q)) \cdot BWs[q]$, to accurately count the loss of credit when q isn't transmitting.

As the difference between 0 and $LRs[q]$ is not negligible anymore, we should only allow the credit to go below $LRs[q]$ when the queue q is not empty. Indeed, the space between 0 and $LRs[q]$ is used to measure the deficit in throughput of q. So, we should not use it when it is empty (i.e., no one is using it). This is implemented at rows [6,7] in 1.

The last added functionality is the one that allows to work with changing capacities. When the instantaneous flow rate of q is below the capacity C, the algorithm works as intended. But the algorithm does not consider when AF instantaneous flow rate is actually above the capacity C (i.e. $dtime[q] < 0$, the next packet is sent before it should be). As shown in line #14 from Algorithm 1: if $dtime[q] < 0$, we add credit in the same way that we decrease credit when $dtime[q] > 0$: we take the increase in the flow rate into account.

This mechanism works because it mimics a Token Bucket Filter, with the only difference is that it increases on average with a slope of $C \cdot (1 - BW)$.

2.4 Defining PSS parameters from WRR weights

The architecture proposed in RFC5865 [3] proposes to lay on a Weighted Round Robin (WRR) scheduler or a variant to share the traffic between AF and CS0 flows[2]. The rationale is that compared to WFQ, WRR provides quite similar guarantees with an easier implementation [22]. Finally, WRR can be implemented both in hardware and software, while WFQ can only be implemented in software due to the virtual clock algorithm [22]. As said previously, the PSS has three parameters: L_M, L_R, and BW.

In the context of wired packet switched networks, where the outgoing capacity is known and fixed, L_R must be set to zero. We will see later the importance of choosing $L_R > 0$ when the capacity of the link varies in Section 2.3. For instance, over satellite link, cellular link or any kind of varying wireless link.

[2]Even for software-based routers, WRR seems preferred. See *QoS Scheduling and Queuing on the CISCO Catalyst 3550 Switches.*, configuration manual available online `http://www.cisco.com/c/en/us/support/docs/lan-switching/lan-quality-of-service/24057-187.html`

To be compliant with the standard architecture, we seek to set L_M and BW to get the same characteristic than a WRR in terms of offered capacity. We call W_i the weight of class $i \in \{AF, CS0\}$, which is a weight corresponding to the number of consecutive packets that can be sent. As WRR is upstream to the priority scheduler, it shares the residual capacity left by the EF traffic between AF and CS0. This capacity is equal to $C - R_{EF}$. As a result, the theoretical output rate of the AF class denoted $R^{*,th}_{WRR/AF}$ is:

$$R^{*,th}_{WRR/AF} = K_{AF} \cdot (C - R_{EF}) \tag{1}$$

with K_{AF} defined in (2), the relative weight of the AF class, which is the share given by WRR to AF relatively to the share given to CS0, with L^{avg}_i the average length of a packet of class $i \in \{AF, CS0\}$.

$$K_{AF} = \frac{W_{AF} \cdot L^{avg}_{AF}}{W_{AF} \cdot L^{avg}_{AF} + W_{CS0} \cdot L^{avg}_{CS0}} \tag{2}$$

We call sending window a period when AF traffic is continuously transmitting, and idle window a period between two sending windows. Our aim here is to share the residual bandwidth left by the EF traffic, in a core network router. So, we have several hypotheses: 1) the presence of EF traffic is not considered when computing the windows, but will be considered for the reserved bandwidth, 2) in a core router, the AF traffic is made of aggregated flows saturating the allocated bandwidth (the rate of a TCP flow increases until a bottleneck is reached), 3) the residual bandwidth available to CS0 flows is insufficient resulting in the fact that CS0 traffic is always available. From the definition of the PSS and our hypothesis, we deduce that in the presence of both AF and CS0 traffics, the minimal sending window is the time needed by the credit to increase from 0 to L_M: $\Delta^{min}_{send} = \frac{L_M}{I_{send}}$, and the minimal idle window is the time needed by the credit to decrease from L_M to 0: $\Delta^{min}_{idle} = \frac{L_M}{I_{idle}}$. Due to non-preemption, an additional packet can be sent if its transmission started just before the credit reached L_M or 0. So the maximal sending window is given by:

$$\Delta^{max}_{send} = \frac{L_M}{I_{send}} + \frac{L^{max}_{AF}}{C},$$

and the maximal idle window by:

$$\Delta^{max}_{idle} = \frac{L_M}{I_{idle}} + \frac{L^{max}_{CS0}}{C}.$$

From these above equations, we can deduce that if we define N such as at most $N-1$ packets of average size L^{avg}_{AF} can be fully sent during Δ^{min}_{send}, then, we have:
$(N-1) \cdot \frac{L^{avg}_{AF}}{C} \leqslant \Delta^{min}_{send} < N \cdot \frac{L^{avg}_{AF}}{C} \leqslant \Delta^{max}_{send}$.
Due to non-preemption, N packets of average size will be sent during an average sending window. The same remains true for the idle window.

During the transmission of the non-preempted packets, the credit saturates either at L_M or $L_R = 0$, and no credit is gained or lost during this time. Thus, the credit is only aware of traffic sent during minimal sending and idle windows. Since the credit is responsible for maintaining the used capacity at BW, we compute BW considering only $N-1$ packets when we want to send N packets. To obtain a PSS service equivalent to WRR, we need to send W_{AF} packets during an average sending window, and W_{CS0} packets during an average idle window. To achieve this, we include the EF traffic to compute the PSS parameter BW as follows:

$$BW = B \cdot (C - R_{EF}), \tag{3}$$

with:

$$B = \frac{(W_{AF} - 1) \cdot L^{avg}_{AF}}{(W_{AF} - 1) \cdot L^{avg}_{AF} + (W_{CS0} - 1) \cdot L^{avg}_{CS0}} \tag{4}$$

Finally, since we seek to send W_{AF} packets of average size during an average sending window, we set L_M so that $W_{AF} - 1$ packets can be sent during the minimum sending windows. Thus, W_{AF} packets will be sent in an average sending window:

$$\begin{aligned} \Delta^{min}_{send} &= \frac{L_M}{I_{send}} = \frac{L^{avg}_{AF}}{C} \cdot (W_{AF} - 1) \\ L_M &= L^{avg}_{AF} \cdot (W_{AF} - 1) \cdot (1 - BW) \end{aligned} \tag{5}$$

We have presented the PSS algorithm and showed how to compute PSS parameters following those obtained by WRR parameters. We now propose to simulate and verify the consistency of the PSS parameters in the presence of a constant bit rate (CBR) UDP traffic. Finally, we will illustrate the advantage of the PSS in the presence of dynamic UDP traffic.

3 Experiments hypothesis

We validated the proposed mechanism over a Mininet virtual network connecting two virtual hosts: one sender and one receiver.

3.1 Implementation

To better assess the performance of this proposal, we implemented and compared PSS to DWRR within a TUN/TAP UDP tunnel. This approach easily allows to experiment these scheduling schemes without complex kernel implementation. The only overhead is the UDP encapsulation. Basically, incoming traffic is received from a local `tun` interface, then incoming packets are scheduled following the algorithm tested, and finally sent to the network encapsulated in the UDP tunnel. The same procedure applies on the server side in the reverse order, where received packets are simply decapsulated and transmitted to the local `tun` interface. We recall that a practical implementation of PSS is freely available to reproduce our experiments[3].

3.2 Test conditions

The common parameters for all the experiments are the following:

- the capacity of the link between the two hosts is limited to 20Mbit/s;
- RTT is set to 500ms to represent a GEO link;
- for each experiment, the traffic generation is done with iPerf[4]. For EF traffic, it consists of a single CBR UDP connection while AF and CS0 traffics are each generated with 10 concurrent CUBIC [14] flows;
- we measure each second the throughput of each flow and the evolution of the credit when PSS is used;
- each run lasts 100 seconds;

4 Experimental results and analysis

This experimental section is sliced in two parts: we first verify and confirm with real measurements, the results previously obtained in [12] by simulation with ns-2 for a fixed capacity and then, we experiment with variable capacity the enhanced PSS algorithm proposed in this paper.

4.1 Experiments with a fixed capacity and EF traffic

4.1.1 PSS and DWRR when EF traffic is fixed (traffic known)

The objective of this section is to show that under the same known and constant UDP/EF load, both PSS and DWRR have the same behavior. To achieve this, we study the service obtained in terms of output rate. Knowing the EF traffic, we set the parameters as explained above with the residual service that corresponds to $C - R_{EF}$ where C is the full capacity of the link. In this experiment, we vary two parameters: the percentage of capacity allocated by the DWRR to the AF class: K_{AF}, and the number of EF flows, which modifies the EF input rate R_{EF}.

Concerning AF and as explained in 2.4, the DWRR shares between AF and CS0 classes the residual capacity left by the EF traffic. The capacity allocated to AF is defined in (4). For a tuple (W_{AF}, W_{CS0}), when R_{EF} increases, the capacity allocated to AF decreases. Fig. 5(a) confirms this theoretical behavior. As PSS curves fit DWRR ones, we can deduce that both schemes have the same behavior. It follows that:

$$R^{*,sim}_{PSS/AF} = R^{*,sim}_{WRR/AF} = R^{*,th}_{WRR/AF}.$$

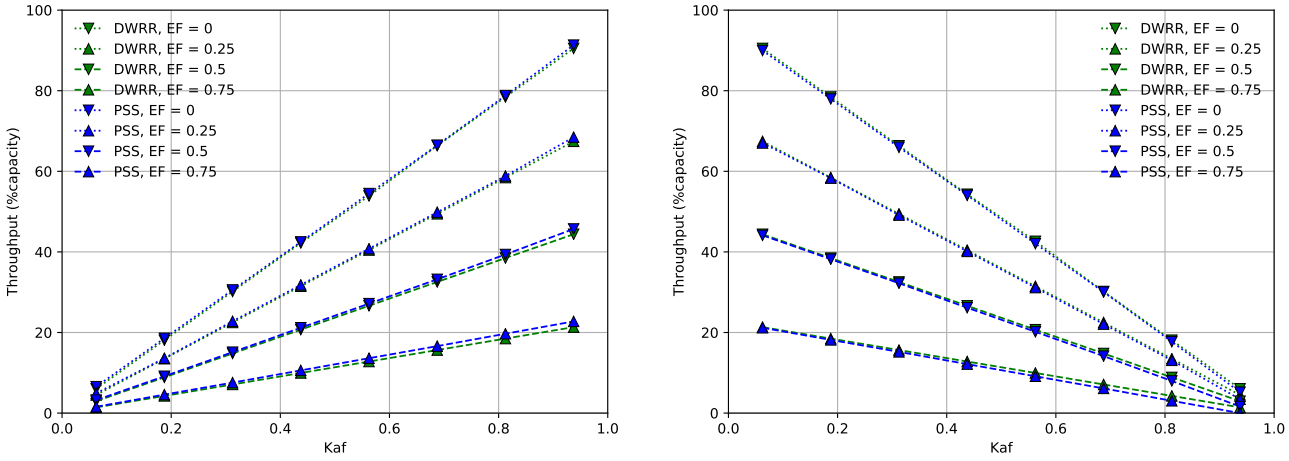
Concerning CS0, CS0 flows share the remaining capacity allocated by the WRR:

$$R^{*,th}_{WRR/CS0} = (1 - K_{AF}) \cdot (C - R_{EF}), \tag{6}$$

with K_{AF} already defined in (2). In Fig. 5(b), we can see the expected behavior again confirmed.

This part shows that with a simple translation of the WRR parameters and when EF traffic is known, the PSS gets similar performance for both the EF and AF traffics. We now have to assess whether the PSS achieves a better isolation of the AF flows than the WRR when EF traffic is unknown.

[3] `https://github.com/elochin/priority-switching-scheduler`
[4] `https://iperf.fr/`



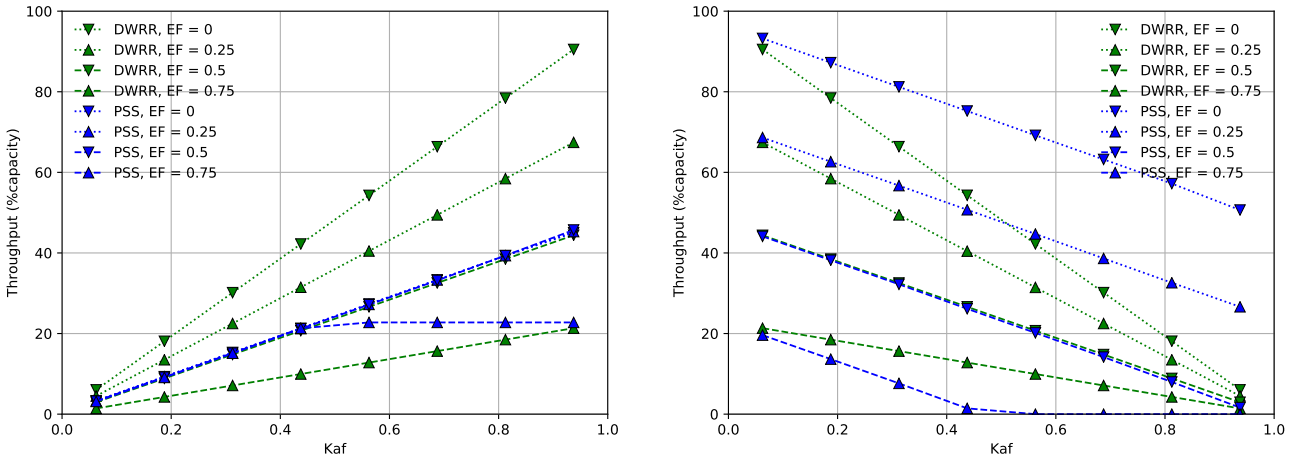
(a) AF class throughput

(b) CS0 class throughput

Figure 5: PSS behaves just like DWRR when C and EF throughput are as expected

4.1.2 PSS and DWRR when EF traffic is varying (traffic is unknown)

In this experiment, R_{EF} is set to 50% while changing the effective EF traffic percentage.



(a) AF class throughput

(b) CS0 class throughput

Figure 6: The error margins of PSS are smaller with an expected $EF_{avg} = 0.5C$

In the previous section, we showed by setting PSS parameters following (3), that PSS and WRR get a similar behavior when EF traffic is known. We now consider that the PSS parameters are set for a certain amount of expected EF traffic denoted R_{EF}^{exp}. So, $BW = B \cdot (C - R_{EF}^{exp})$, and L_M can be computed with (5). Thus, theoretical PSS output rate for the AF class is:

$$R_{PSS/AF}^{*,th} = \min\left[K_{AF} \cdot (C - R_{EF}^{exp}), C - R_{EF}\right] \tag{7}$$

The *min* is done between the residual capacity and the theoretical AF output rate as given by (1). This is linked to the network stability condition, i.e., the total input traffic rate has to be lower than the transmission capacity of the scheduler, of the AF traffic.

Concerning CS0 traffic, it uses the remaining capacity left by the EF and AF traffics. With the necessary stability condition due to AF priority being lower than EF, the resulting theoretical PSS output rate for CS0 class is:

$$
\begin{aligned}
R^{*,th}_{PSS/CS0} &= C - R_{EF} - R^{*,th}_{PSS/AF} \\
&= \max\left[C - R_{EF} - K_{AF} \cdot (C - R^{exp}_{EF}), 0\right] \\
&= \max[(1 - K_{AF}) \cdot (C - R_{EF}) \\
&\quad + K_{AF} \cdot (R^{exp}_{EF} - R_{EF}), 0] \\
&= \max[R^{*,th}_{WRR/CS0} + K_{AF} \cdot (R^{exp}_{EF} - R_{EF}), 0]
\end{aligned} \tag{8}
$$

We now have theoretical output rates for AF and CS0 with PSS in (7) and (8) resp. with WRR in (1) and (6). We will use these results to interpret the simulations.

Following a study published in 2015 [21], the part of the Internet real-time traffic is ranging from 35 to 65%. So, we consider an expected rate of R^{exp}_{EF} at 50% of the capacity C. We vary the EF traffic to estimate the performance of the PSS. The WRR weights do not depend on the EF traffic, so the results of our simulations with the WRR are identical to Fig. 5(a). The results in Fig. 6(a) confirm the theoretical output rate calculations:

$$R^{*,sim}_{PSS/j} = R^{*,th}_{PSS/j}, j \in \{AF, CS0\}.$$

We observe while WRR rates are strongly impacted by the EF traffic, with the PSS, as long as the stability limit is not reached, the AF class has the same output rate when EF varies. Of course, if the AF traffic is not impacted, then this means the impact is fully on the CS0 class, as defined in (8). This is visible in Fig. 6(b). The results in Fig. 6(a) and Fig. 6(b) can be linked to theory as follows:

1. When $R_{EF} = 0.75 \cdot C$, we have $R_{EF} > R^{exp}_{EF} = 0.50 \cdot C$. As long as $R^{*,sim}_{PSS/AF} \leqslant C - R_{EF}$, then $R^{*,sim}_{PSS/AF}$ increases as defined by the relation $K_{AF} \cdot (C - R^{exp}_{EF})$. Then, when $K_{AF} \cdot (C - R^{exp}_{EF}) \geqslant C - R_{EF}$, the output $R^{*,sim}_{PSS/AF}$ is limited by the priority scheduler due to the EF traffic. This leaves $R^{*,sim}_{PSS/AF} = C - R_{EF}$. So $R^{*,sim}_{PSS/AF} = R^{*,th}_{PSS/AF}$. When we compare $R^{*,sim}_{PSS/AF}$ to $R^{*,sim}_{WRR/AF}$ we find in Fig. 6(a) that $R^{*,sim}_{WRR/AF} \leqslant R^{*,sim}_{PSS/AF}$. This fits the theoretical output rates, since we have $K_{AF} \cdot (C - R_{EF}) < K_{AF} \cdot (C - R^{exp}_{EF})$ and $K_{AF} \cdot (C - R_{EF}) \leqslant C - R_{EF}$. The same analysis can be done for CS0 traffic, proving that $R^{*,sim}_{PSS/CS0} = R^{*,th}_{PSS/CS0}$ and $R^{*,th}_{WRR/CS0} \geqslant R^{*,th}_{PSS/CS0}$ which fits the results on Fig 6(b). Briefly, when $R_{EF} = 0.75 \cdot C$, we have R_{EF} higher than the one expected. The AF rate is lower with DWRR than with the PSS because DWRR shares the reduced capacity (25% instead of 50%) between the AF and EF classes. In particular, it shows that the rate obtained for AF with DWRR is lower than the rate obtained by PSS.

2. when $R_{EF} = 0.50 \cdot C$, we are in the case where R_{EF} obtained is the one expected. So, the DWRR and the PSS have the same output rates as they were designed to, when R^{exp}_{EF} was set to 50% of C;

3. when $R_{EF} = 0.25 \cdot C$, we have R_{EF} lower than the one expected. The AF rate is higher with DWRR than with the PSS because DWRR shares the benefit of the additional capacity (75% instead of 50%) between AF and EF classes. The same analysis as we did for $R_{EF} = 0.75 \cdot C$ can be done in this case. It shows that the rate obtained for CS0 is lower than the rate obtained by PSS, while the rate obtained for AF with DWRR is higher than the rate obtained by PSS.

To sum up, this confirms the results obtained with ns-2 simulation in [12]: PSS effectively limits the impact of UDP/EF variations on the TCP/AF traffic.

4.2 Experiment results with changing capacity and EF Throughput

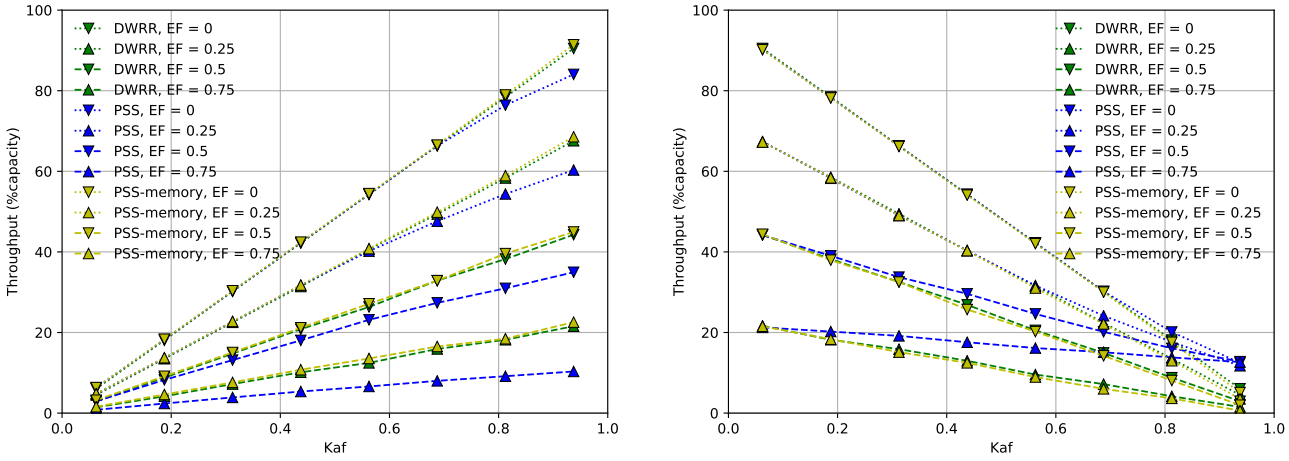
As described in 2.3, a satellite link might have a varying capacity. Furthermore, although some admission control can be enabled, EF flow rate might not be constant. To test the robustness of the proposed mechanism considering these characteristics, we introduced a moving capacity and a moving EF throughput both as a function of time as follows:

- $C(t) = C_{avg}.(1 + 0.3\cos(\frac{2\pi t}{15}))$
- $EF(t) = EF_{avg}.(1 + 0.6\cos(\frac{2\pi t}{6.1}))$

The objective is not to specifically reproduce bandwidth variation ($C(t)$ strongly oscillates here) of a real satellite or EF traffic, but to obtain two different period of variation to cover a large kind of cases:

1. an AF class in starvation as the instantaneous capacity available $(C - EF)(t)$ is too low;

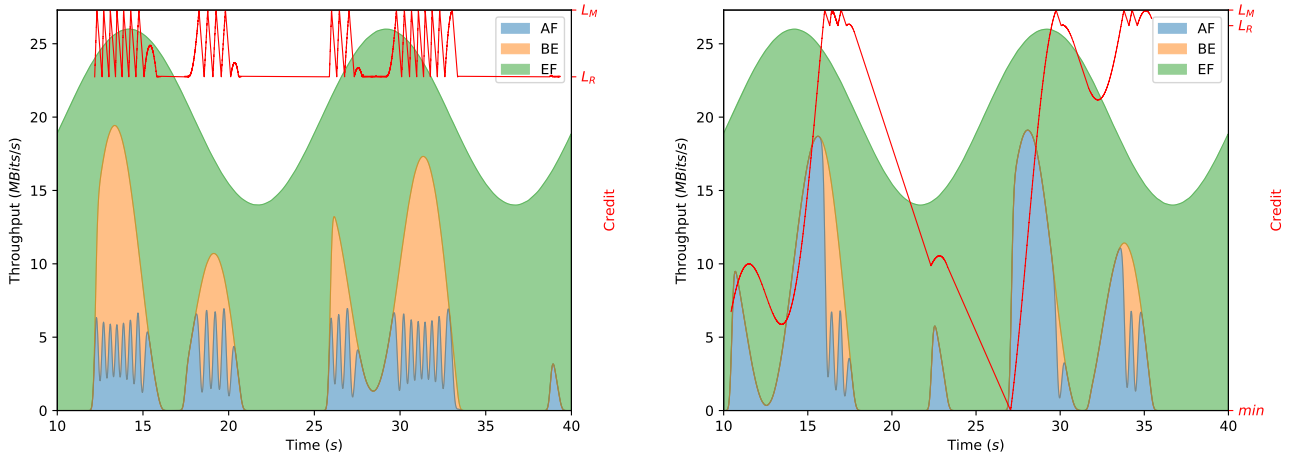
2. an AF traffic that reaches the target and share with BE the remaining capacity.



(a) AF class throughput

(b) CS0 class throughput

Figure 7: Difference between PSS with capability to adapt to varying capacity (with credit memory tracker) and with vanilla PSS (no memory credit tracker)



(a) Evolution without memory

(b) Evolution with memory

Figure 8: Evolution of throughput/credit with PSS with and without memory

4.2.1 Importance of well sizing L_R

As shown in Figures 7, even knowing the mean capacity and EF throughput, the PSS algorithm without memory mechanism is not robust. The guarantees are weaker. This phenomenon is exacerbated when the part allocated to EF becomes higher: when $(C - R_{EF})(t) < (C_{avg} - R_{EF})(t)$. This behavior can be explained with Figures 8(a), 8(b). These figures represent the evolution of throughput of AF, CS0 and EF classes, with $K_{AF} = 0.81$ and $EF_{avg} = 0.75\,C_{avg}$. Figure 8(b) illustrates that going below L_R value allows memorizing starvation period occurring to AF traffic while in Figure 8(a) a share of the remaining capacity is done between AF and BE to AF.

How to set L_R?

L_R should be sized as a function of the variability of the capacity of the link. For instance, the starvation period can be determined following the standard deviation of the link capacity or a threshold value can be chosen following the service desired.

Basically :

1. if L_R is chosen too small, the capability to memorize starvation period is also small. PSS might also behave as if no memory was set. If the link capacity C or the EF traffic are highly variable and cause AF traffic rate to be lower than the target: $BW.C_{avg}$ over certain periods, in the long term the AF traffic might not achieve the target;

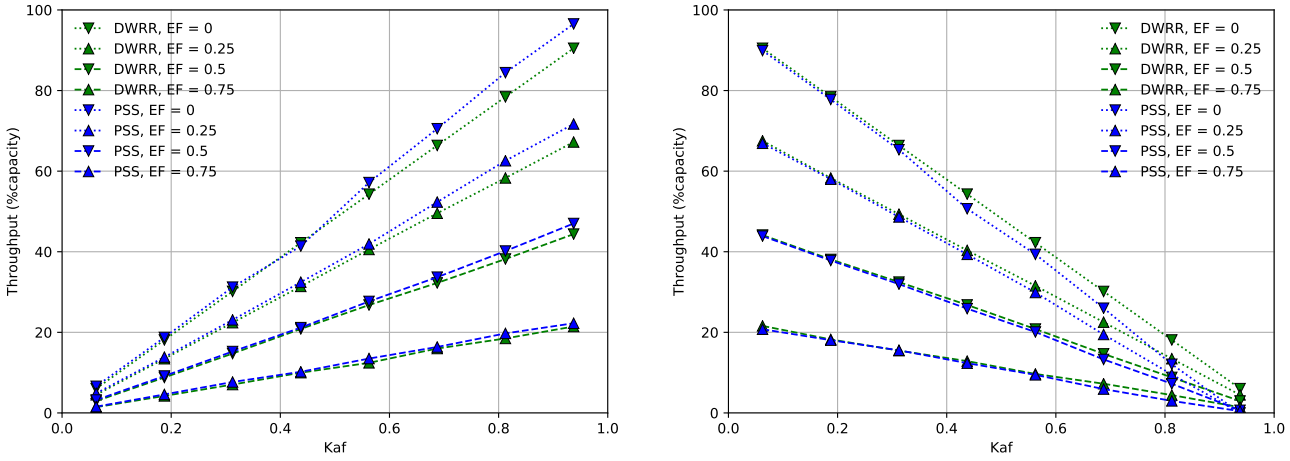
2. if L_R is too large, the history memorized will allow AF traffic to compensate the gap every time possible, to the price of a potential starvation of CS0 traffic.

In other words, L_R must be chosen considering:

1. the evolution of both the link capacity C and EF traffic. For example: let's consider C periodic over $T\,sec$, then a fair value is: $L_R = T.C_{avg}.BW$. Indeed, at worse the available capacity is below $C_{avg}.BW$ for $T\,sec$, thus allowing PSS to memorize $T\,sec$ of complete starvation;

2. the policy of the operator, depending on what kind of service it provides. L_R is a guarantee of how much data the scheduler is allowed to transmit later, while keeping in sight the guarantee targeted $C_{avg}.BW$ and the CS0 service the operator seeks to provide.

4.2.2 Considering a greater capacity



(a) AF class throughput

(b) CS0 class throughput

Figure 9: Case where PSS is not considering a higher capacity

In the left Figure 9, PSS mechanism uses more than the target capacity ($BW.C_{avg}$), especially when the available capacity is bigger, thus penalizing CS0 flows. We are now in the case where $C(t) > C_{avg}$.

As explained in 2.3, we need to consider that the instantaneous flow rate may at some moment (like $t = 15s$ and $t = 30s$ in figures 10(a) and 10(b)) be superior to C_{avg}. The difference is illustrated in figures 10(a) and 10(b). Indeed, we can see that when $C(t) > C_{avg}$, the first algorithm gives more than its fair share to the AF class, leaving CS0 with an even lower share.

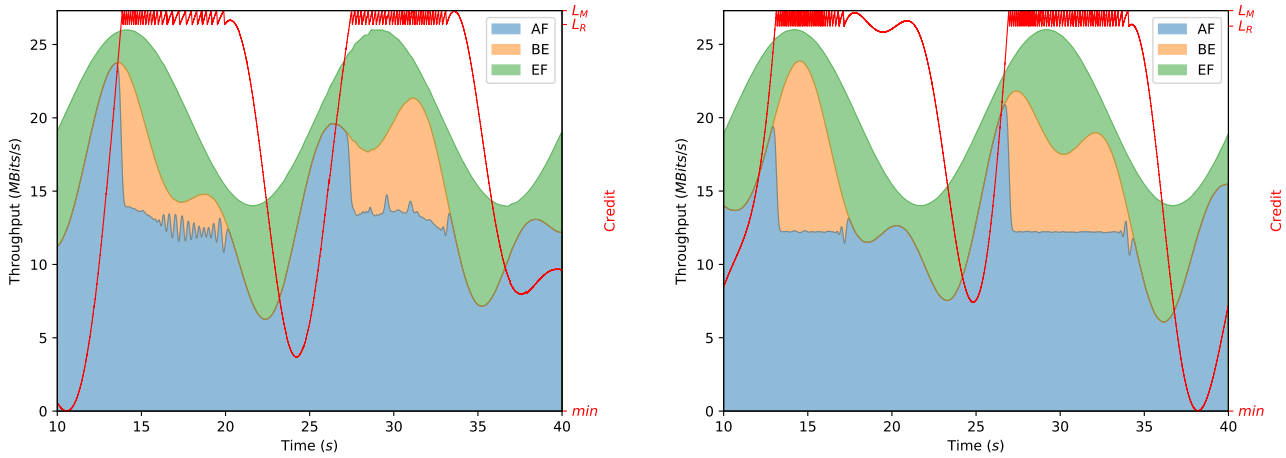
In 10(b) we can observe that as soon as AF class regained all of its credit, the throughput is immediately going back to target $C.BW$ throughput.

4.2.3 Aggregation

Just like in 4.1, we verify in this section if PSS has the same behavior as DWRR if the average capacity and EF throughput are accurately predicted, and then we will see how the margins of error holds when EF is not as expected. In these case, the capacity and EF throughput are variable to consider the variability of a satellite link, as explained in 4.2. The results obtained are shown in Figure 11 when C and EF throughput are as expected for $EF_{avg} = 0.5C$.

5 Discussion

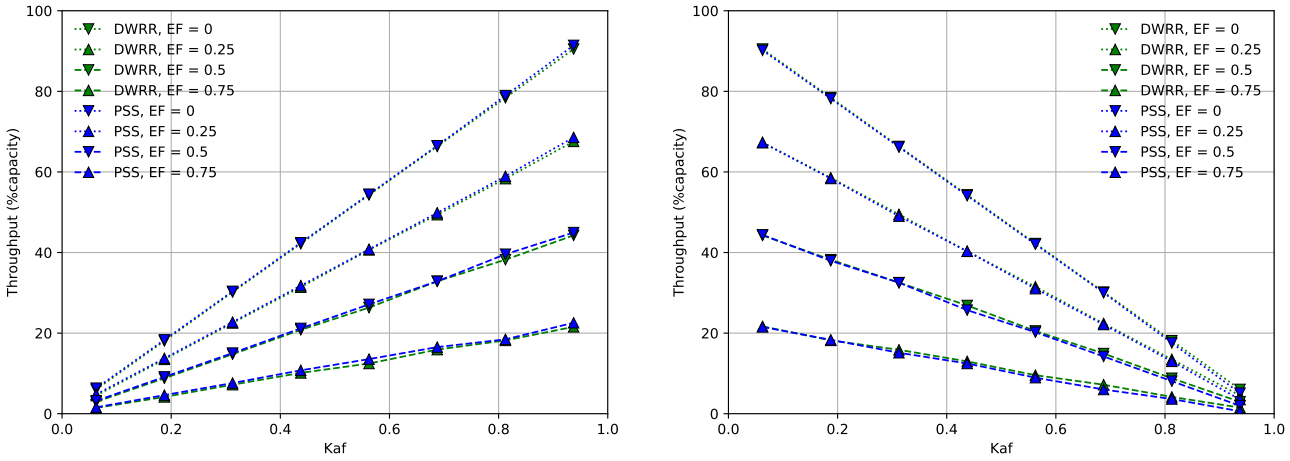
To ease the understanding of the experimental results presented and clearly assess the properties of the Priority Switching Scheduler, we illustrate in Figure 13 on the left: the theoretical output rate of a WRR rate scheduler as proposed in the RFC5865 [3], based on the analytical development in Section 4; with the AF output rate obtained by PSS on the right. As shown, the AF class has a more predictable available capacity, while the unpredictability is reported on the CS0 class. With good parameters setting, both classes also have a minimum assured rate.



(a) Evolution of throughput/credit with PSS

(b) Evolution of throughput/credit with PSS when taking into account an higher capacity

Figure 10: Evolution of throughput/credit considering or not a higher capacity



(a) AF class throughput

(b) CS0 class throughput

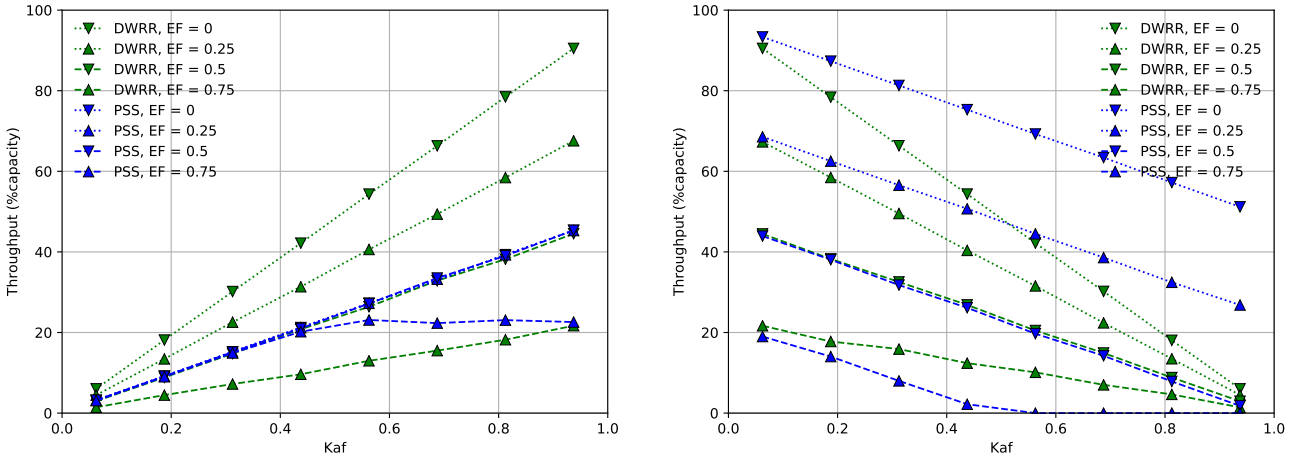
Figure 11: PSS behaves just like DWRR when C and EF throughput are as expected.

These theoretical results presented in figure 13 are supported by real measurements in Figures 6(a) and 12(a).

As last but not least, PSS objective is to efficiently quantify the reserved capacity for the AF class and to improve the AF service while EF class remains managed by a priority scheduler. As a matter of fact, PSS is strongly dependent on throughput variations than on transmission time variations. The variation of the goodput (useful bitrate) is mainly due to the efficiency of the MODCODs used in GEO. The capacity oscillation scenario from section 4.2 used to stress PSS (ranging from $15Mb/s$ to $25Mb/s$) demonstrates that PSS remains robust to high variation in capacity. Although this study focuses on GEO system, we believe the measurements presented should hold in a context of NGSO systems where capacity variations might also occur [19]. A perspective of this work could be to design an adequate tested and measurement sets to verify the performance of PSS over such a system.

6 Conclusion

We have proposed a novel core scheduler architecture which better quantifies the performance of the AF class of a core network scheduling. Basically, PSS enforces the rate guarantee offered by the AF class, whatever the traffic in the EF class. In particular, we have shown that the new service offered to the AF class is defined by the relation (7). Compared to the WRR, the AF output rate is less dependent on the EF traffic, which



(a) AF class throughput

(b) CS0 class throughput

Figure 12: PSS error margins are smaller with expected $EF_{avg} = 0.5C$.

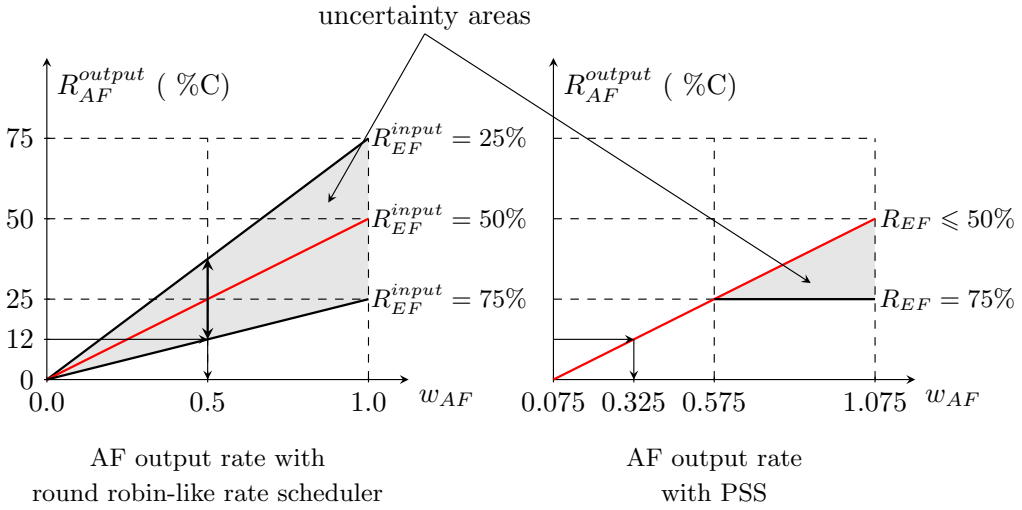


Figure 13: Representation of the gain obtained between standard rate schedulers and PSS

improves the quantification of the reserved capacity of AF, without impacting EF traffic. Real measurements illustrate the new kind of service offered to the AF class by PSS. Compared to the DWRR, the AF output rate is less dependent on the EF traffic, which improves the quantification of the reserved capacity of AF, without impacting EF traffic. Furthermore, the enhanced version proposed makes PSS more robust when the available capacity of the link is changing over the time. Finally, we have confirmed the findings in [12] and illustrated that a correspondence between DWRR weights and PSS parameters can be achieved, making this proposal simple to deploy by network engineers.

References

[1] Toufik Ahmed, Emmanuel Dubois, Jean-Baptiste Dupé, Ramon Ferrús, Patrick Gélard, and Nicolas Kuhn. Software-defined satellite cloud ran. *International Journal of Satellite Communications and Networking*, 36(1):108–133, 2018.

[2] Eitan Altman and Tania Jiménez. Morgan and Claypool publishers, 2012.

[3] F. Baker, J. Polk, and M. Dolly. A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic. RFC 5865 (Proposed Standard), May 2010.

[4] B. Bensaou, K. T. Chan, and D. H. K. Tsang. Credit-based fair queueing (cbfq): a simple and feasible scheduling algorithm for packet networks. In *IEEE ATM Workshop 1997. Proceedings*, pages 589–594, May 1997.

[5] Igor Bisio, Mario Marchese, and Maurizio Mongelli. Performance enhanced proxy solutions for satellite networks: State of the art, protocol stack and possible interfaces. In Kandeepan Sithamparanathan and Mario Marchese, editors, *Personal Satellite Services*, pages 61–67, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[6] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC 3135 (Informational), June 2001.

[7] Torsten Braun, Thomas Staub, Michel Diaz, and José Enríquez-Gabeiras. In *End-to-End Quality of Service Over Heterogeneous Networks End-to-End Quality of Service Over Heterogeneous Networks*. Springer-Verlag Berlin Heidelberg, Berlin, Allemagne, 2008.

[8] Carlo Caini, Rosario Firrincieli, and Daniele Lacamera. Pepsal: A performance enhancing proxy for tcp satellite connections [internetworking and resource management in satellite systems series]. *IEEE Aerospace and Electronic Systems Magazine*, 22(8):B9–B16, 2007.

[9] Gorry Fairhurst, Arjuna Sathiaseelan, Haitham Cruickshank, and Cedric Baudoin. Transport challenges facing a next-generation hybrid satellite internet. *International Journal of Satellite Communications and Networking*, 29(3):249–268, April 2011.

[10] Gorry Fairhurst, Raffaello Secchi, and Ana Yun. A flexible qos architecture for dvb-rcs2. *International Journal of Satellite Communications and Networking*, 31(5):219–232, May 2013.

[11] David Pradas Fernandez. *Cross-Layer Design and Methodology for Satellite Broadband Networking*. PhD thesis, 2011. Universitat Autònoma de Barcelona (UAB) and ISAE-SUPAERO.

[12] A. Finzi, E. Lochin, A. Mifdaoui, and F. Frances. Improving rfc5865 core network scheduling with a burst limiting shaper. In *IEEE Global Communications Conference*, pages 1–6, 2017. IEEE Global Communications Conference.

[13] Franz-Josef Gotz. Traffic Shaper for Control Data Traffic (CDT). *IEEE 802 AVB Meeting*, July 2012.

[14] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: A new tcp-friendly high-speed tcp variant. 42(5):64–74, jul 2008.

[15] Manolis Katevenis, Stefanos Sidiropoulos, and Costas Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose atm switch chip. *IEEE Journal on selected Areas in Communications*, 9(8):1265–1279, 1991.

[16] Sastri L. Kota, Kaveh Pahlavan, and Pentti A. Leppänen. In *Broadband Satellite Communications for Internet Access*. Springer-Verlag Berlin Heidelberg, Berlin, Allemagne, 2004.

[17] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *IEEE/ACM Transactions on Networking*, 12(4):681–693, 2004.

[18] Emmanuel Lochin and Pascal Anelli. TCP throughput guarantee in the DiffServ Assured Forwarding service: what about the results ? *Annals of Telecommunications*, vol. 6(3-4), April 2009.

[19] François Michel, Martino Trevisan, Danilo Giordano, and Olivier Bonaventure. A first look at starlink performance. IMC '22, page 130–136, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Christian Niephaus, Mathias Kretschmer, and Gheorghita Ghinea. Qos provisioning in converged satellite and terrestrial networks: A survey of the state-of-the-art. *IEEE Communications Surveys Tutorials*, 18(4):2415–2441, 2016.

[21] Sandvine. Global Internet Phenomena Report. Technical report, 2015.

[22] Chuck Semeria. Supporting Differentiated Service Classes: Queue Scheduling Disciplines, 2001. White Paper 200020-001 12/01 - Juniper Networks, Inc.

[23] Shuang Xu, Xingwei Wang, and Min Huang. Modular and deep qoe/qos mapping for multimedia services over satellite networks. *International Journal of Communication Systems*, 31(17):e3793, 2018. e3793 dac.3793.

Acknowledgements

The authors wish to thank Ahlem Mifdaoui (ISAE-SUPAERO), David Baker (IETF) and Nicolas Kuhn (CNES) for numerous discussions on this scheme.