Does the integration of CBS and GCL behaves as you expect? And can it be enhanced?

Hugo Daigmorte, Marc Boyer

Does the integration of CBS and GCL behaves as you expect? And can it be enhanced?

Hugo Daigmorte
ISAE-SUPAERO/University of Toulouse
Toulouse, France

Marc Boyer
ONERA, The French Aerospace Lab
Toulouse, France

December 20, 2018

Abstract

Time Sensitive Networking (TSN) working group has added a set of mechanisms to Ethernet in order to provide a real-time network. In particular, the output port scheduling based on Credit-Based Shaper (CBS) has been enhanced with a time driven Gate Control List (GCL). In this paper, we show that the interactions between both mechanisms can lead to unexpected behaviours, creating bursts of low priority flows. We then propose a small modification of the rules whose aim is to reduce these bursts.

Contents

1 Introduction

Audio-Video Bridging (AVB) is a set of extensions of the IEEE 802.1Q standard, initially given as a set of addenda, and currently integrated in the 802.1Q standard itself.

Among the 8 classes, the two with higher priority (called A and B) are scheduled at the output port of each switch with a specific policy: at configuration, each class $X \in \{A, B\}$ receives a part of the output port rate R, defined as $\text{idleSlope}_X \in [0, R]$. This scheduling, combining a Credit-Based Shaping (CBS) and static priority (SP), provides four properties:

P1: each class X is guaranteed to receive a long term service of rate idleSlope_X,

P2: each class X is limited to a long term service of at most idleSlope_X,

P3: the scheduler limits the bursts of the classes A and B, *i.e.* the waiting time of a frame of the class B will be smaller with an AVB scheduler than with a simple static priority scheduler,

P4: if $\text{idleSlope}_A + \text{idleSlope}_B < R$, the other classes are guaranteed to receive some bandwidth (at least $R - (\text{idleSlope}_A + \text{idleSlope}_B)$).

The property P1 can be called the *minimal throughput* property, the property P2 is a *maximal throughput* property, the property P3 is a kind of *fairness* property, and property P4 is just a consequence of P2.

The Time Sensitive Networking (TSN) working group has added new extensions. One among others, named "Enhancement for scheduled traffic", introduced per class gates: each class has access to the output port only when its gate is open. And the opening/closing of the gates is controlled by a cyclic time based schedule, called *Gate Control List* (GCL). The introduction of this mechanism is interpreted by some authors as the introduction of Time-Triggered communications in TSN. The CBS algorithm has been adapted to take into account the GCL mechanism.

Our claim is that this integration breaks the property P3, creating bursts of higher priority flows and so creating large window without any access to the output for lower priority flows.

For sake of simplicity, this paper does not address preemption.

2 Credit evolution rules with GCL

In the current 802.1Q standard, it must exists at each output port a *Transmission Selection Algorithm* [1][§8.6.8]. The Credit-Based Shaper (CBS, [1][§8.6.8.2]) is the one considered here.

The selection rules of CBS relies, for each queue X, on the value of an integer value, called *credit* and a parameter, idleSlope_X, the *idle slope*. The rules, when no GCL is present, are the following:

R1: The head of queue frame can be selected for emission only if the credit of this queue is non negative, and no higher priority frame is ready for transmission.

R2: During frame transmission, the credit decreases as a function of time with rate $\text{sendSlope}_X = \text{idleSlope}_X - R$ where R is the transmission rate of the port.

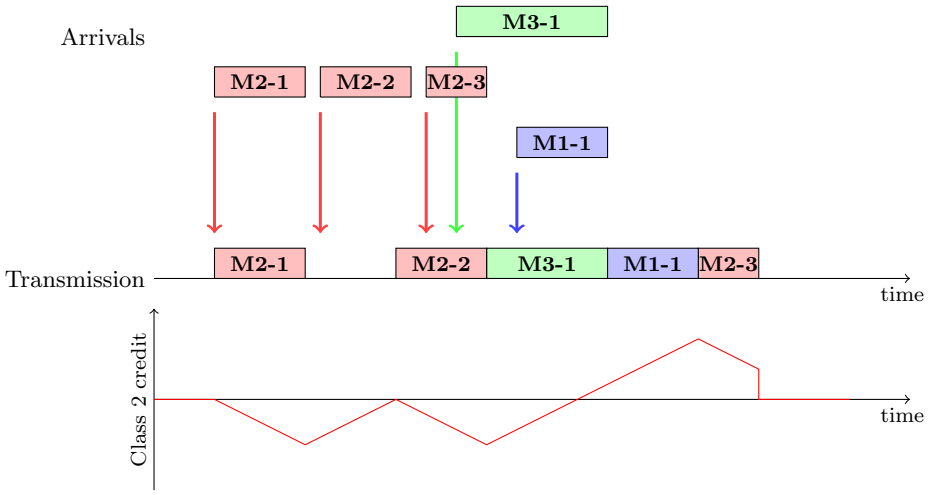

Figure 1: CBS credit evolution rules (without GCL)

R3: The credit increases with rate idleSlope$_X$ when either the credit is negative, or it is positive and there is some frame waiting in the queue.

R4: If the credit is positive and the queue is empty, the credit is reset to null.

This behaviour is illustrated in Figure 1, considering the the evolution of credit of class 2: the initial value is null. When a message `M2-1` is set in the queue, the credit is null, and from rule R1, it can be selected for transmission. Then, rule R2 implies that the credit decreases. When message `M2-2` is received, it can not be selected since the credit is negative, and it must H: wait until the credit of class 2 reach 0 to be selected. At the end of `M2-2` transmission, the output port is idle, and a lower priority message `M3-1` can start its transmission. Since we do not consider preemption, when the credit of class 2 reaches 0, the message can not be send, and its value increases to positive values. However at end of `M3-1` transmission, a higher priority frame `M1-1` is waiting, and `M2-3` has to wait. Thus its credit still increases.

Things are getting slightly more complex when gates are used. In [1][§8.6.8.4] is introduced the notion of gate: to each queue is associated a gate, and this gate can be either closed or open. When a gate is closed, the frame in the associated queue can not be selected for transmission. A static global cyclic table, the GCL, schedules the gate openings and closings.

How does these two mechanisms, CBS and GCL interact? The main ideas are:

1. the credit is frozen when the gate is closed (overriding rules R2 and R3),
2. a frame can starts is emission if it can be sent up to completion before the next closing event.

The key point is: what happens just before gate closing? To prevent a frame to encroach on a closed gate period, a frame can not start its emission if it has not enough time to finish before next gate closing event[1]. What is the evolution rule of the credit during this time? In previous studies [7, 3], this period has been considered has an "extension" of the gate closing period, and it was assumed that the credit was also frozen. But in the current standard [1], credit increases during this waiting time, and it can break some properties of CBS without gate closing.

Since this behaviour may be not obvious for each reader of the standard, we are first going to see which part of the standard is involved in this behaviour.

2.1 Looking into details on credit evolution rules

The credit based shaping algorithm is presented in [1][§8.6.8.2]. The summary presented here is based on the exact words of the standard. The preemption is not considered here for simplicity, and related parts are omitted.

Each queue maintains a boolean variable, *transmit*, that "takes the value true for the duration of a frame transmission from the queue; false when any frame transmission from the queue has completed" [1][§8.6.8.4-e].

It also maintains the *credit* value. One rule states that "If ... there are no frames in the queue, and the transmit parameter is false, and credit is positive, and the transmission gate for the queue is open ... then credit is set to zero" [1][§8.6.8.4-f].

The credit is increasing with rate idleSlope "while transmit is false and the transmission gate for the queue is open" [1][§8.6.8.4-d]. The credit is decreasing with rate sendSlope "while transmit is true" [1][§8.6.8.4-g].

Then, the section dedicated to gates [1][§8.6.8.4] presents the "enhancements for scheduled traffic". A "gate control list" defines a static list of instants, cyclically repeated, defining "gate-close" and "gate-open" events. After a gate-close event, the gate is in the closed state, until the next gate-open event, that will set the gate in the open state.

It also states that "in addition to the other checks carried out by the transmission selection algorithm, a frame on a traffic class queue is not available for transmission if the transmission gate is in the closed state or if there is insufficient time available to transmit the entirety of that frame before the next gate-close event associated with that queue".

Now, the question that we address is: what is the evolution rule of the credit when there is a frame in a queue, the gate is open, but not enough time to send it before the next gate closing?

In such a case, the transmit variable is false (there is no transmission). And since the gate is still in open state the rule "transmit is false and the transmission gate for the queue is open" [1][§8.6.8.4-d] is applied, and the credit variable is increasing...

[1]The use of preemption may reduce this "forbidden" time, but it can not completely avoid it.

CBS	Credit-Based Shaper
$GateOpenTime(n)$	the accumulated opening time of the gate of queue n
GCL	Gate Control List
$MaxPreCloseTime$	Maximum value of pre-closing during a GCL
$OperCycleTime$	the duration of the GCL
$portTransmitRate$	the transmission rate of the port of interest
$operIdleSlope(n)$	bandwidth (in b/s) reserved for queue n
$PreCloseTime(s,t)$	accumulated time of pre-closing between s and t
R	shorthand for $portTransmitRate$

Table 1: Glossary

A small remark: the fourth note of section 8.6.8.2 states that "The highest value that can be accumulated in credit depends on idleSlope and the length of time that the algorithm may have to wait when the queue is not empty and there is other traffic being transmitted through the Port". But the rule that forbids emission of frame to avoid encroaching on the next closed gate period defines a time interval where it may have no traffic being transmitted.

2.2 Credit evolution rules are unfair

This evolution of credit before a gate closing can have unexpected consequences. The reason is that, for a given class, there are two sources of bandwidth loss due to gate closing: some is lost while the gate is closed, and during these intervals, the credit is frozen, whereas some is lost while the gate is open, and during these intervals, the credit increases.

To discuss the differences and interactions between these elements, some vocabulary has to be introduced.

2.2.1 Vocabulary

Let $portTransmitRate$ be the transmission rate of the port of interest. Let $N \in [0,7]$ be a queue number, then $OperCycleTime$ is the duration of the GCL table (the period of the behavior), $GateOpenTime$ the accumulated opening time of the gate of queue N during the GCL table duration, $GateCloseTime = OperCycleTime - GateOpenTime$, the accumulated closing time of the gate. The network designer has to set the $operIdleSlope$ of each queue N using CBS, the bandwidth allocated to this queue [1][§34.3]. All these values are statically defined. Now, given an interval $[s,t]$, $PreCloseTime(s,t)$ is the accumulated time where some frame has been delayed because of rule "there is insufficient time available to transmit the entirety of that frame before the next gate-close event"[2]. This duration is by nature dynamic: it depends on the presence of

[2]We chose the term "pre-closing" time for these intervals, instead of "guard band" which is more commonly used, because depending on the authors, the "guard band" is this dynamic pre-closing time, whereas for others, the "guard band" is a static interval before the closing

frame in the queue before some gate-close event, on the credit value, and on the frame size. It can be upper bounded: each close-gate event may generate only one pre-close interval, and its length is at most the duration of one frame of maximal length. Let *MaxPreCloseTime* be its maximal value on a GCL cycle[3].

Then, the *idleSlope* parameter is computed as [1][§8.6.8.2]

$$idleSlope(N) = operIdleSlope(N) \cdot \frac{OperCycleTime}{GateOpenTime} \quad (1)$$

where $\frac{\text{GateOpenTime}}{\text{OperCycleTime}}$ is the fraction of time the gate is open.

2.2.2 The pre-closing time can not be used by CBS flows

The evolution rules of credit prevent the use of this pre-closing time by CBS flow: first, the static configuration of parameters prevents from statically reserving it for CBS flows, second, the transmission rules prevents CBS flows from dynamically using it.

Pre-closing time can not be allocated to CBS flows At configuration time, the pre-closing time can not be allocated to CBS. That is to say, the sum of allocated bandwidth plus the *GateCloseTime* plus the *MaxPreCloseTime* must be less than the *portTransmitRate*. Otherwise, the value of the credit can overflow.

More precisely, considering a TSN output port, where

- the GCL list is such that, when the gate of the highest priority queue (the one with index 7) is open, all other gates are closed, and when the gate of the highest priority queue is closed, all other gates are open (aka. *exclusive gating*),
- all queues except the highest and lowest priority (the one with index 0) ones have a CBS shaper.

Then, for any priority $n \in [1, 6]$, the condition

$$\sum_{i=6}^{n} \text{operIdleSlope}(i) + \text{GateCloseTime}(n) + \text{MaxPreCloseTime}(n) \leq R \quad (2)$$

must hold, otherwise, the credit of the priority level n can overflow.

An example of overflow is illustrated in Figure 2.

In this example, there are two CBS flows: class A and class B. The evolution of their credits is represented at the top of the figures.

---

event, whose length must be large enough to prevent any encroaching. In the standard itself, the term "guard band" is used only in an informative appendix [1][Appendix Q], and without any exact definition.

[3]The terms *portTransmitRate*, *OperCycleTime*, *GateOpenTime operIdleSlope*, are from the standard, the terms *GateCloseTime*, *PreCloseTime* and *MaxPreCloseTime* have been introduced to ease the discussion.

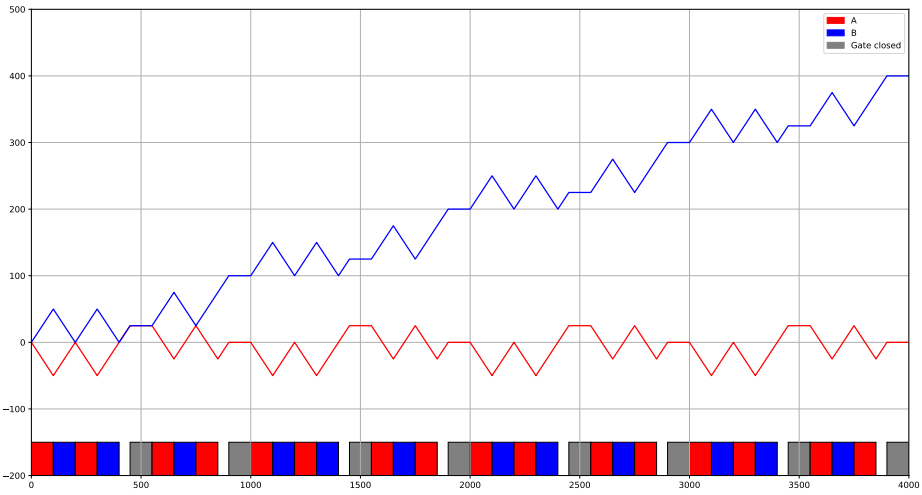

Figure 2: Credit overflow

At the bottom of the figures, is represented the emission of the messages as well as the closing gate period.

The *OperCycleTime* is set equal to 1000, the *portTransmitRate* is also 1000, all messages are of constant sizes 100 and the *GateCloseTime* represent 20% of the *OperCycleTime* divided into two 100 size windows. Both examples assume that the queues are never empty during the considered interval.

In this example the sum of allocated bandwidth plus the *GateCloseTime* is equal to *portTransmitRate* (class A and class B *operIdleSlope* equal 400). But since a part of the bandwidth allocated is in fact lost by pre-closing time, it leads to credit overflow.

Un-allocated time can not be used by CBS flows The previous example has shown that the pre-closing time, which is by nature dynamic, can not be allocated to CBS flows. One may wonder if it can be dynamically used by these flows.

Consider a second example (see figure 3) with the same parameters than the previous one, except that the sum of allocated bandwidth plus the *GateCloseTime* plus the *MaxPreCloseTime* is equal to *portTransmitRate* (class A and class B *operIdleSlope* equal 300).

In this case, there is no more overflow; however there is some unused bandwidth. The reason is that the idle slope of a CBS queues acts both as a minimal reserved capacity, but also as a maximal bandwidth usage (it is both the slope of the minimal service curve and the shaping curve [5]). Then, since the *MaxPreCloseTime* has not been allocated to CBS flows (to avoid possible credit overflow), it can not be used by these CBS flow.

This behaviour is not very surprising: whereas GPS-like policies (WFQ, WRR, DRR [4, 2, 6]) are designed to *dynamically* share the bandwidth in a fair

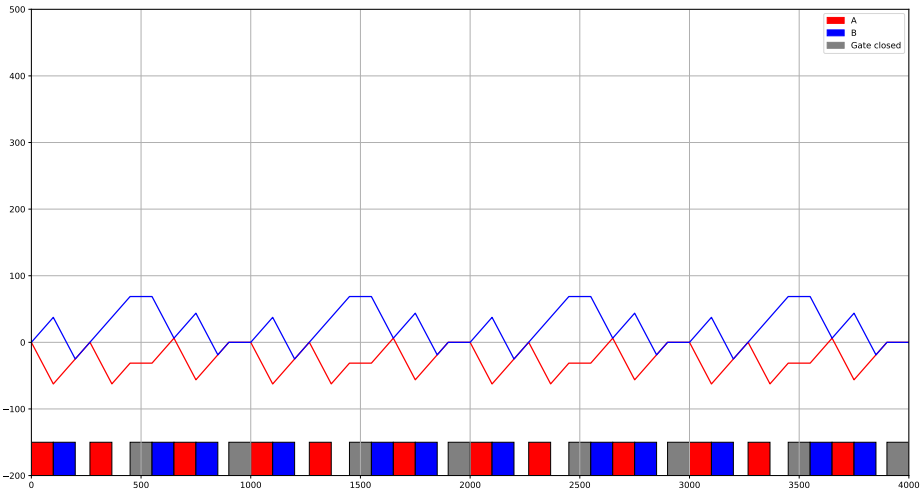

Figure 3: Unused bandwidth

way (with configurable weights between flows), credit based shaping reserves a *constant* capacity. In TSN, to avoid encroaching on static gate closing, some dynamic loss of bandwidth is created, and credit based shaping can not share it between flows (but it may be used by some Best-Effort queue, *i.e.* queue with low priority flow and without shaper).

2.2.3 The pre-closing time credit evolution rule is unfair

A more unexpected consequence of the credit evolution rule is that it is unfair: it will allow high priority classes to accumulate more credit, and create larger burst than what would append without gate closing. It then increases the worst waiting time of low priority frames.

In order to visualise this unfair behaviour we will use the original situation describe in the figure 4. In these examples there are three CBS flows : class A, class B and class C. In this example the *GateCloseTime* is null and the *portTransmitRate* is 1000. 30% of the *OperCycleTime* is allocated to class A (*operIdleSlope* equal 300), 20% is allocated to class B (*operIdleSlope* equal 200) and 20% is allocated to class C (*operIdleSlope* equal 200). Messages of each class are supposed to be of constant sizes : 150 for A and 100 for B and C. The figure 4 shows that there is an alternation in the transmission of messages between these three classes.

Now we will consider a different situation. One third of the class A traffic is changed to *Time-Triggered* traffic: its *operIdleSlope* is decreased to 200, its message size is decreased to 100. The remaining part is cut into smaller packets of size 50, send to the higher priority queue, with a schedule given at bottom of Figure 6 ([4]). Like in the previous case, we assume that when this queue gate is

[4]One may object that it is not possible to have Ethernet frames of size 50, and that this

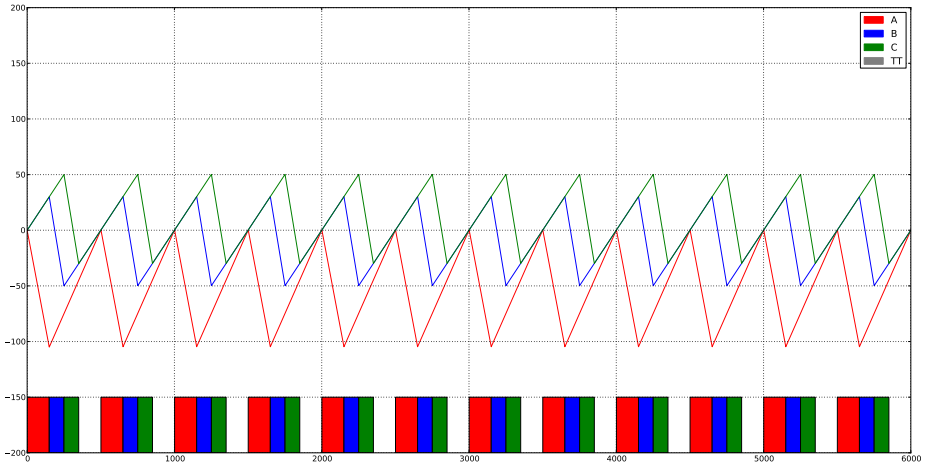

Figure 4: Three CBS flows

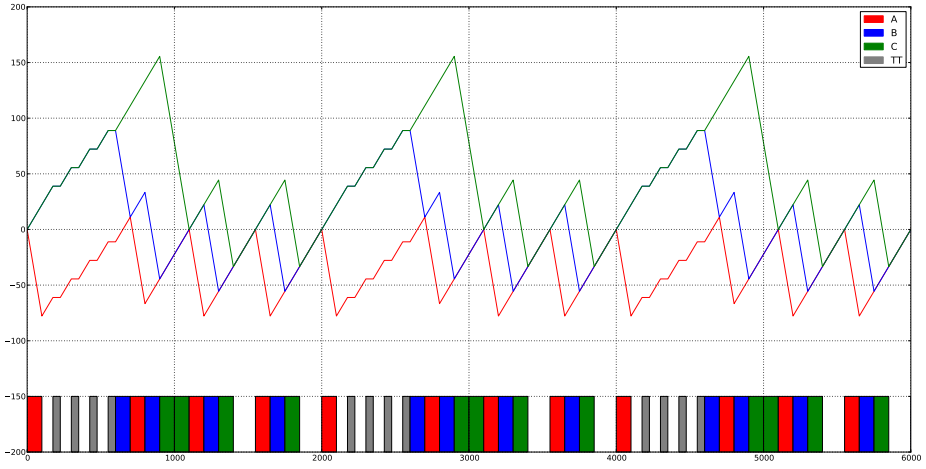

Figure 5: Credits increase during the pre-closing time

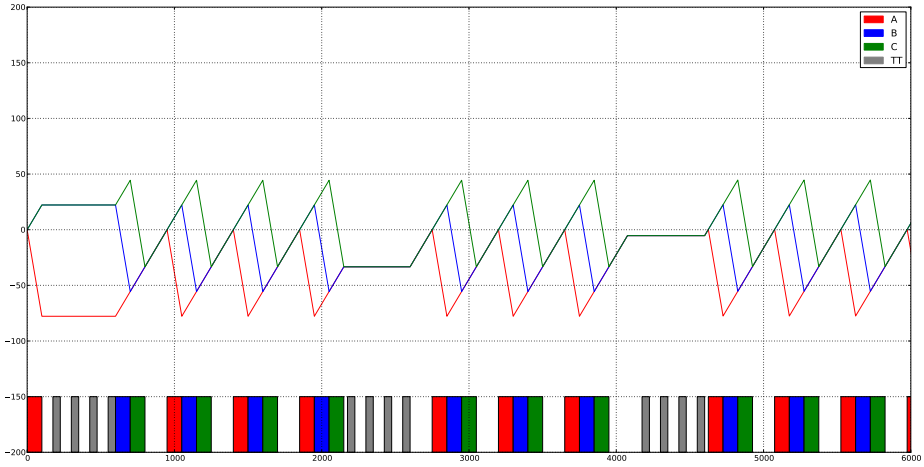

Figure 6: Credits are frozen during the pre-closing time

open, all other are closed, and conversely.

Then, the behaviour of the system is represented in figure 5: the first frame of the C queue is delayed by two frames from A and two frames from B. The long term service is the same (in both Figures 4 and 5, 12 frames of C are served), but the traffic is more bursty, due to gate closing, but also to credit evolution rule.

2.2.4 But this can be fixed

We propose a small modification of credit evolution rule: when a frame is delayed *due to a next gate close event* (during a *PreCloseTime*), the credit is frozen (like in a gate-close period). This is a small modification: it still is a local decision, and just move forward one state change.

The Figure 6 represents the same system than in Figure 5, with our rule modification: credit freeze during pre-closing. In this situation there is a perfect alternation in the transmission of messages between the three classes. This situation is more fair.

However the part of unused bandwidth increases: since the credits do no more increase during *PreCloseTime*, only 10 frames per class are served during this schedule. The reason is in the conversion between the *operIdleSlope* and the *idleSlope*: the eq. 1 does not consider the effect of pre-closing.

Now, let *MaxPreCloseTime* be, for a given class, the maximum time lost due to pre-closing during a full GCL table: for each closing event c_i, its duration is either the maximal transmission time of a frame of this class ($\frac{L^{\max}}{R}$, with $L^{\max}$ the maximal length of a frame), or the distance with the previous open gate even (in case the opening interval is not large enough to allow the transmission

schedule with very small intervals is a very non efficient one. But this example has been build to illustrate some behaviour with human-friendly constants.

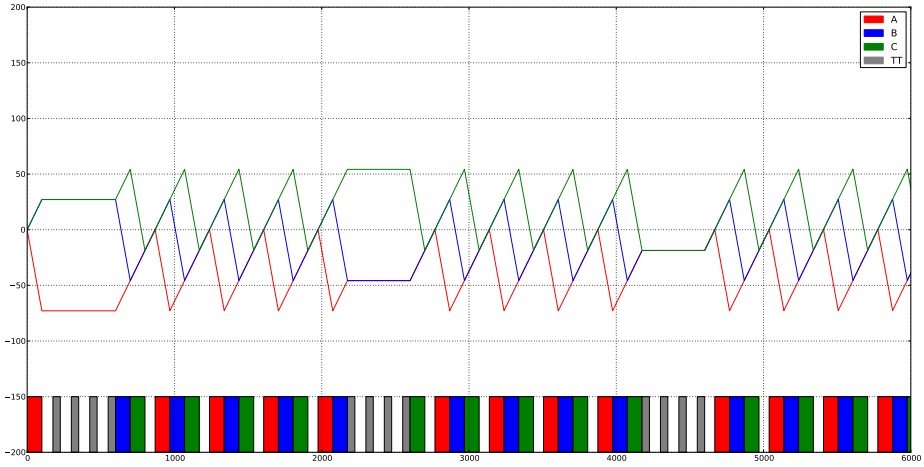

Figure 7: Credits are frozen during the pre-closing time, idle slopes computed using eq. 3

of a maximal length frame). Then, the idle slope of each class can be computed as:

$$idleSlope(N) = operIdleSlope(N) \cdot \frac{OperCycleTime}{GateOpenTime - MaxPreCloseTime} \tag{3}$$

$$MaxPreCloseTime = \sum_{c_i \in \texttt{GCL}} \min\left((c_i - o_{i-1}), \frac{L^{\max}}{R}\right) \tag{4}$$

where `GCL` is the sequence of closing time, and o_{i-1} the "previous" opening time.

With this enhanced rule, the systems behaviour is still a perfect alternation between classes of the 3 classes, and the bandwidth usage is maximal.

3 Conclusion

This article pays attention to a small part of the TSN behaviour: the instant before a gate close event, and their interactions with the CBS selection algorithm, and in particular the credit evolution rules before a gate-closing event.

First, it shows that the behaviour described in the current version of the standard [1] is different from what was assumed in several previous studies [7, 3]: it continues to evolves whereas others assume its is frozen.

Second, it recalls that, due to encroaching avoidance, this part of the bandwidth can be lost. It can be dynamically used by some best-effort flow, but this part of the bandwidth can not be allocated to CBS queues, otherwise, is may lead to credit overflow.

Third, this paper shows on an example that the standard rule can increase the burstiness of the system.

Then we analyse the reasons of this burstiness, and propose a small modification that can make the scheduling more smooth. It is sufficient to freeze the credit before gate closing, like assumed in [7, 3], and to slightly change the way the operator idle slope parameter is transformed into the queue idle slope.

Thanks We would like to thank Luxi Zhao from Technical University of Denmark for the valuables comments on a preliminary version of this paper.

References

[1] IEEE standard for local and metropolitan area networks – bridges and bridged networks. IEEE Standard 802.1Q, IEEE, 2018.

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

[3] Dorin Maxim and Ye-Qiong Song. Delay Analysis of AVB traffic in Time-Sensitive Networks (TSN). In *RTNS 2017 - International Conference on Real-Time Networks and Systems*, page 10, Grenoble, France, October 2017.

[4] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking (TON)*, 1(3), June 1993.

[5] Joan Adrià Ruiz de Azua and Marc Boyer. Complete modelling of AVB in network calculus framework. In *Proc. of the 22nd Int. Conf. on Real-Time Networks and Systems (RTNS 2014)*, Versailles, France, October 8-10 2014.

[6] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. *SIGCOMM Computer Communication*, 25:231–242, October 1995.

[7] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. Timing analysis of avb traffic in tsn networks using network calculus. In *Proc. of the 23th IEEE Real-Time and Embedded Technology and App. Symp. (RTAS 2017)*, pages 25–36, Pittsburgh, USA, 2017. IEEE.