Equivalence between the theoretical model and the standard algorithm of Asynchronous Traffic Shaping

Marc Boyer

Equivalence between the Urgency Based Shaper and Asynchronous Traffic Shaping in Time Sensitive Networking

Marc Boyer
ONERA/DTIS, Université de Toulouse, F-31055 Toulouse, France

Abstract

The Asynchronous Traffic Shaping (ATS) has been designed by the Time Sensitive Networking (TSN) group as a reshaping mechanism for real-time data flows, based on the initial proposition of the Urgency Based Shaper (UBS). Several studies have exhibited properties and limitations of this solution, but most of them are based on the model presented in the UBS definition [20], whereas the implementation described in the standard uses a different architecture and algorithm. This paper presents an equivalence proof between the model and the standard specification.

2012 ACM Subject Classification Networks → Formal specifications; Networks → Packet-switching networks; Networks → Cyber-physical networks; Networks → Traffic engineering algorithms

Keywords and phrases TSN, Time Sensitive Networking, ATS, Asynchronous Traffic Shaping, 802.1Qcr

Digital Object Identifier 10.4230/LITES.9.1.1

Received 2022-09-26 **Accepted** 2023-12-14 **Published** 2024-04-08

1 Introduction

The Time Sensitive Networking (TSN) group of IEEE aims at defining addenda to extend Ethernet with real-time properties (this extended Ethernet is commonly called TSN). This is done, among other things, by adapting or defining new arbitration policies in output ports: the credit-based shaper (CBS, [1]), the Enhancements for Scheduled Traffic (sometime called "Time Aware Shaper", [7]) and last the Asynchronous Traffic Shaping (ATS, [4]), which is the main subject of this paper.

The initial proposition was called "Urgency-Based Scheduler" (UBS, [20]). It defines a new scheduling algorithm where flows are re-shaped after reception by a switch before being put in the output queue. Two kind of reshaping are considered, the Length Rate Quotient (LRQ) and the Token Bucket Emulation (TBE). This re-shaping requires specific queues (called "shaped queues" to store frames that need to be delayed), and some active elements (called "regulators") in charge of deciding when each head of queue can be forwarded to the output queue. BS has a very strong property, despite the introduction of delay for some frames in the reshaping phase, it does not introduce any "additional delay" in the worst case [20, § IV.C] (see Section 4 for details). This property is commonly know as "reshaping for free" [14].

The TBE version of the mechanism has been standardised as a TSN addendum, under the name "Asynchronous Traffic Shaping" (ATS, [4]), but with quite a different architecture. Instead of storing frames and deciding if a frame must be forwarded to the output queue or delayed when it becomes the head of queue, an algorithm computes, when the frame is received, an "eligibility time". The frame is then forwarded to the output queue, but it will be eligible for transmission only after its eligibility time.

Then, a lot of studies have been done on this mechanism [14, 23, 28, 16], but although most claim to model ATS, they rely on the UBS description from [20]. This paper proves that the mechanism specified in the standard model [7] is, up to the reordering of frames with the

same eligibility time, equivalent to the TBE version of UBS from [20]. It also shows that some implementation variability allowed by the standard (the place of some sub-component in the switch) has an impact on the interactions between ATS and the Frame Replication and Elimination for Reliability mechanism (FRER, [2]),

In the rest of the paper, the term UBS denotes the TBE version of UBS as described in [20], and the term ATS denotes the mechanism specified in the standard model [7].

The paper is organised as follows. The first part presents the context: Section 2 presents UBS, Section 3 presents ATS (with a specific subsection on the integration of UBS in TSN, Section 3.5), and Section 4 presents the related work. The second part presents the contribution: the equivalence between UBS and ATS in Section 5, and the interaction between ATS and FRER positions in TSN architecture in Section 6. Section 7 concludes.

2 Presentation of UBS

This section presents UBS, as defined in [20] (up to some renaming to ease readability, and considering only the TBE shaping mechanism). It starts with a reminder on the notion of token bucket in Section 2.1. Then, UBS is presented as the assembly of different elements. Section 2.2 presents the "shaped queue", a queue that is shared by different token buckets, leading to the name of "interleaved shaping" (that has been generalised under the name "interleaved regulator" in [14]). Different shaped queues can be assembled into an "UBS queing system", as presented in Section 2.3 (this naming was not in [20] but its introduction eases the presentation). Last, Section 3.5 shows how such system was designed to be integrated into an UBS switch, and gives a list of requirements for the flow to queue allocations.

Note that this bottom-up presentation is not the one used in [20]. In [20], the presentation starts from an Ethernet switch with several priority levels enhanced with a set of *shaped queues* (also called "mandatory" queues), some others queues (one per priority level), called "pseudo-queues" or "optional queues", and the assembly emerges from an adequate per priority-level mapping.

2.1 Recall on the token bucket

The token bucket is a well known mechanism in network [27] but since there exist several flavors, and no reference naming, here is proposed a terminology (a discussion takes place in Section 4).

A token bucket is a system with two parameters: a capacity, also known as burst, b (in some data unit, e.g. bits or frame) and a replenishment rate r (in data unit per time unit). The state of the bucket is the number of tokens it contains. There are three main evolution rules.

TB1 The bucket is initially full.

TB2 The bucket is continuously replenished at rate r, but is limited to its capacity b.

TB3 When a frame of size s is forwarded, the bucket is decremented either by one (if the data unit is the frame) or by the size of the frame (if the data unit is the byte, the bit, or an equivalent unit).

From theses common rules, three kinds of token-bucket-based system can be defined:

- traffic-shaping token bucket (sTB): Such a system also involves a queue, and it is designed to regulate (or shape) a flow. If there are not enough tokens (this number can be 1 if the token unit is the frame, or the frame size in bits, bytes, etc.), the frame is delayed until the bucket is replenished with enough tokens. The output is then "shaped" or "regulated". This is illustrated in Figure 1.

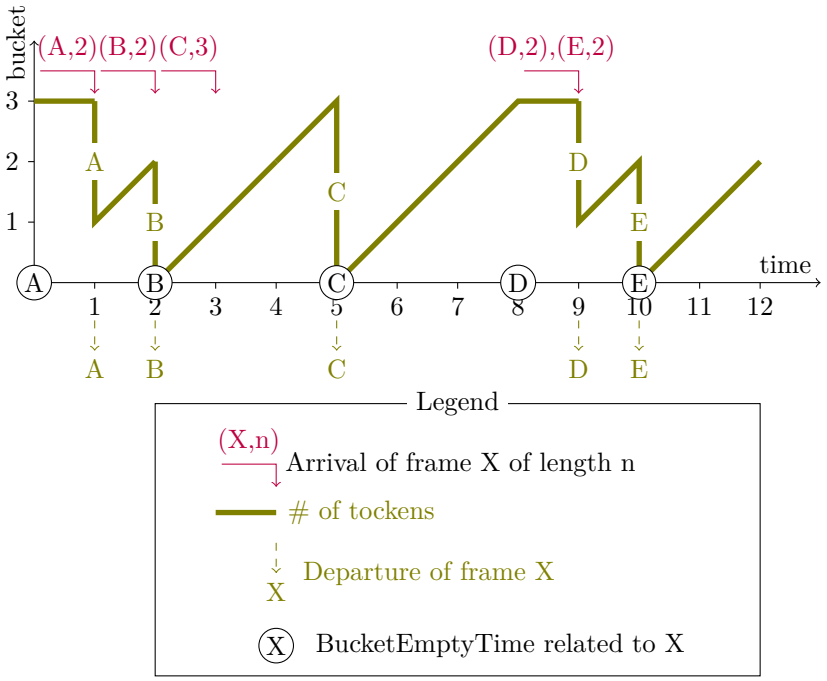

Figure 1 Illustration of the behaviour of a traffic-shaping token bucket with $r = 1$, $b = 3$. – The `BucketEmptyTime` will be defined in Section 3.4 – The bucket is initially full. When the frame A of size 2 is received at time 1, there are enough tokens, the frame is delivered and the bucket is decreased by 2. Then, it is replenished up to time 2, at reception of frame B, that goes through, consuming all tokens. When frame C is received at time 3, there is only 1 token in the bucket, whereas the size of C is 3. Then, C is delayed up to time 5. The plateau between times 8 and 9 shows that the replenishment is bounded by the bucket capacity. Then two frames, D and E, are received at the same instant, but there are enough tokens only for the first one, and the second is delayed.

- policing token bucket (pTB): In policing token bucket, there is no queue. When a frame arrives, if there are not enough tokens, the frame is dropped. If there are enough tokens, the number of tokens is updated as for a sTB.
- classifier token bucket (cTB): In a classifier, there is no queue. When a frame arrives, if there are not enough tokens, the frame is marked as "out of contract" with a specific tag (a drop tag, a red or orange colour [15],...), and forwarded without modification of the bucket state. If there are enough tokens, the frame is marked with a "nominal" tag (a green colour [15]) and the number of tokens is updated as for a sTB.

The TBE shaping in UBS relies on traffic-shaping token bucket.

2.2 Shaped queue and interleaved shaping

We first present UBS "in isolation", i.e. as a mechanism regulating a set of flows, without any considerations on routing, switching or TSN.

Let us first consider a sub-part, called the "shaped queue" [20, § III.A].

Consider a set of m flows, that we will name "group", $G = \{f_1, f_2, \ldots, f_m\}$, a single queue q (that will be called the "shaped queue"), and a set of m token buckets, the i-th having parameters (r_i, b_i). The specific point is that all token buckets share the queue q. This architecture in isolation is illustrated in Figure 2 with $m = 3$.

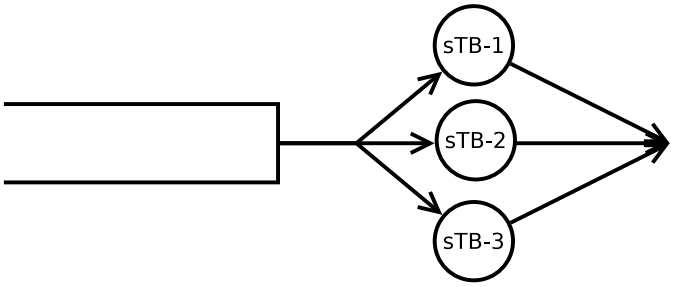

Figure 2 Shaped queue: example of a single queue with three token buckets.

The behaviour of the system is the following. Each flow deposits packets in the queue. The queue is handled in a FIFO way. Each token bucket maintains its own number of tokens, applying the rules of a traffic-shaping token bucket. When a frame reaches the head of queue, the system identifies to which flow it belongs (using a "stream identification function" [2]). Then, it asks to the associated token bucket if they are enough tokens. If yes, the frame is forwarded, applying rule TB3. If not, the frame is delayed up to point in time when the bucket is replenished with enough tokens, like a traffic-shaping token bucket. The specific point is that the rest of the queue is also delayed (due to FIFO rule), even if the next frame has enough tokens in its own bucket. Keep in mind that if there is always at most one token bucket handling the head of queue, the others continue to be replenished during this time.

An example of behaviour is illustrated in Figure 3.

This systems where several shapers share the same queue is called "interleaved shaping" [20, §III.C], and generalised as "interleaved regulators" in [14].

2.3 UBS queuing system

The next system is based on the assembly of several shaped queues.

Consider a set of n groups, $G_1, \ldots, G_n$, each group G_i being a set of m_i flows, i.e. $G_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,m_i}\}$, sharing a *shaped queue* q_i with token bucket parameters $r_{i,j}, b_{i,j}$. Each flow belongs to only one queue ($\forall i, j : G_i \cap G_j = \emptyset$). To this system is also associated a queue, that we call the "ready queue", the output of all token buckets, as illustrated in Figure 4.

We found no explicit name for such a assembly of one ready queue and its associated shaped queues, so we call it an "UBS queuing system".

The first presentation of UBS, in [20], suggests to have an implementation different from the full UBS model. It suggests that the implementation does not require a ready queue (and the ready queue is called "pseudo-queue" in [20]), but only add to ready frames a time tag, that stores the instant when this frame has been accepted for forwarding by its queuing token bucket. Then, instead of selecting the head of queue of the ready queue, one may simply select the head of queue of the shaped queues with the smallest ready time tag.

We will not give more details on this simplification here. The interested reader may find details in [20].

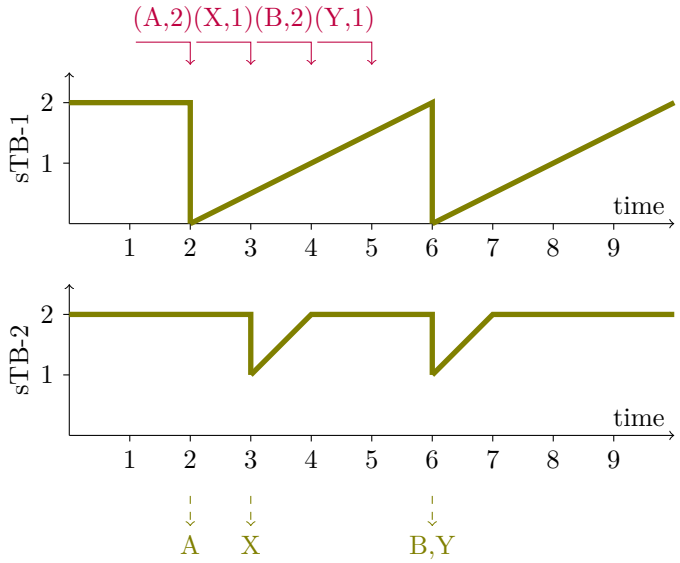

Figure 3 Illustration of behaviour of an interleaved shaper with two token bucket shapers, sTB-1 and sTB-2 with $r_1 = 1/2$, $r_2 = 1$, $b_1 = b_2 = 2$, the first one shaping frames A, B from flow 1, and the second shaping frames X, Y from flow 2. Both buckets are initially full. When the frame A of size 2 is received at time 2, there are enough tokens is sTB-1, the frame is delivered and the bucket is decreased by 2. Then, it is replenished at rate 1/2. The frame X of size 1 is received at time 2, there are enough tokens in sTB-2 bucket, the frame is delivered and the bucket is decreased by 1. This illustrates the fact that each flow uses its own bucket. When the frame B of size 2 is received at time 4, there are not enough tokens in sTB-1, the frame is delayed. But when the frame Y of size 1 is received at time 5, there are enough tokens in sTB-2, but Y is not the head of queue. Then, Y has to wait until the departure of B (at time 6). At this instant, the scheduler can test its bucket, consume one token and deliver Y.

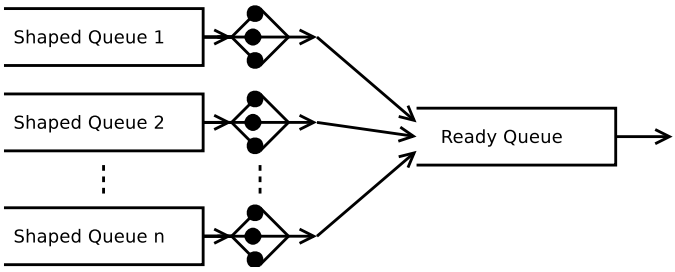

Figure 4 An UBS "queuing system": several shaped queues and one ready queue.

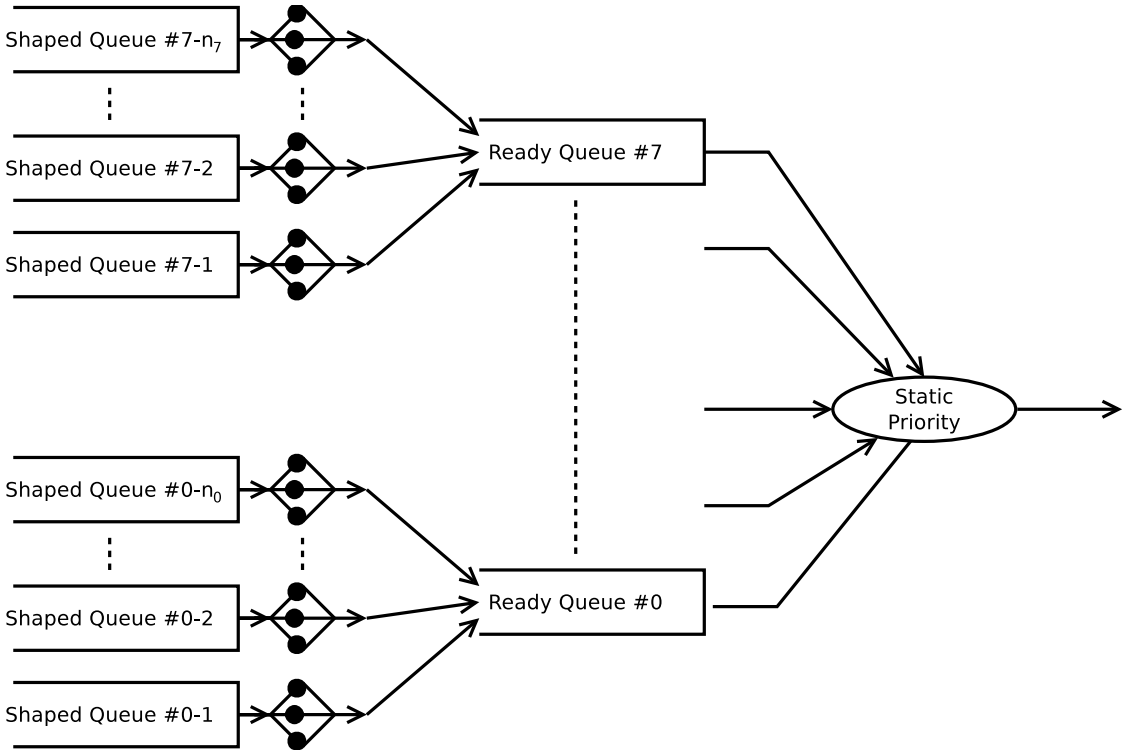

Figure 5 Architecture of a UBS output port (with 8 ready queues, each ready queue of priority i having n_i shaped queues).

2.4 An UBS output port

In an Ethernet or TSN switch, an Ethernet frame can have a priority level, in 0..7 (7 being the highest priority), and there are 8 traffic classes[1]. To each class, in each output port, is associated a queue.

The UBS proposal consists in adding a dedicated set of shaped queues before each queue, per priority level, as illustrated in Figure 5 (with renaming the Ethernet "queues" to "ready queues").

To get the full benefit of the mechanism (i.e. a "reshaping for free", the fact that the delay introduced by the interleaved regulator does not increase the worst delay), some segregation between flows in queues must be respected [20], [14]. In fact, three conditions are given in [20, § III.B]:

QAR1 [two frames] are not allowed to share a [shaped] queue if both are received from different switches,

QAR2 [two frames] are not allowed to share a queue if both are sent by the same switch in different priority levels,

QAR3 [two frames] are not allowed to share a queue if both are sent by [the local switch] in different priority levels.

In fact, if a switch has n (input) ports, if all switches in the network have m priority levels ($m \leq 8$ in Ethernet), each output port can receive frames from at most n switches. If moreover, each switch does not change the priority level of the frame, the set of constraints QAR1–QAR3

[1] In fact, it may exist less, but for sake of simplicity, we keep this assumption here.

can be trivially satisfied with n shaped queue per ready queue, one ready queue per priority level (i.e. in Figure 5, $\forall j \in 0 \dots m : n_j = n$), requiring mn shaped queues, storing frames from input port i with priority level p in the shaped queue p-i.

In this case, an UBS output port requires mn shaped queues plus m ready queues.

But for routing reasons, some shaped queues may be unused (if they are only n_j switches sending data with priority j, then n_j shaped queues are sufficient.

In the general case, the allocation of shaped queues to ready queues and data flows must respect constraints QAR1–QAR3.

So, one drawback of UBS is that it may require a large number of queues (up to $m(n+1)$). As presented at end of Section 2.3, it is suggested in [20] that an implementation may not use the "ready queues" (then called "pseudo-queues") if the static priority scheduler is able to directly look in shaped queues, using only up to mn queues. As will be presented in next section, ATS proposes an "equivalent" mechanism without any shaped queues, using only m ready queues (plus some eligibility time mechanism).

3 Presentation of ATS

3.1 TSN output port (pre-ATS)

Let first present the architecture of a TSN output port (depicted in Figure 6), before the introduction of ATS.

In a TSN switch, an Ethernet frame can have a priority level, in 0..7 (7 being the highest priority), and there are 8 traffic classes[2]. To each class, in each output port, is associated a "queuing system"[3].

To each traffic class is associated a "Transmission selection algorithm" (TSA), that can be either "Strict priority", "Credit-based shaper" (CBS), "Enhanced Transmission Selection" (ETS), "ATS Transmission Selection" or some vendor specific value [4, Table 8-6]. This TSA is sometimes called a "shaper".

Each traffic class is also controlled by a gate, which is open or closed, depending on the clock value and a configuration table (the Gate Control List – GCL). A static priority arbiter always selects the frame in the highest priority traffic class (if it has a frame ready for transmission).

That is to say, the head of queue is selected for transmission by the output port only if is TSA has released it (each TSA having it own conditions), if the gate allows it (it is open, and there is sufficient time to send the frame before the next closing event), if there is no higher priority class with a frame ready for transmission (including its own TSA and gate conditions), and if there is no lower priority frame using the link (up to some fragmentation condition [6]).

Figure 6 illustrates the architecture of a TSN output port, with 8 traffic classes, without any details on the transmission selection algorithm.

3.2 Insertion of ATS in the forwarding process

In a TSN switch, once a frame is received in a reception port, it goes to an "Active topology enforcement" process, in charge of some routing activities, then to a sequence of filtering processes ("Ingress filtering", "Frame filtering" and "Egress filtering"), mainly in charge of discarding frames

[2] In fact, it may exist less, but for sake of simplicity, we keep this assumption here.

[3] In this report, we will use the term "queuing system" whereas the standard use the term "queue", but with the following note "A queue in this context is not necessarily a single FIFO data structure. A queue is a record of all frames of a given traffic class awaiting transmission on a given Bridge Port." [4, § 8.6.6, Note 3]

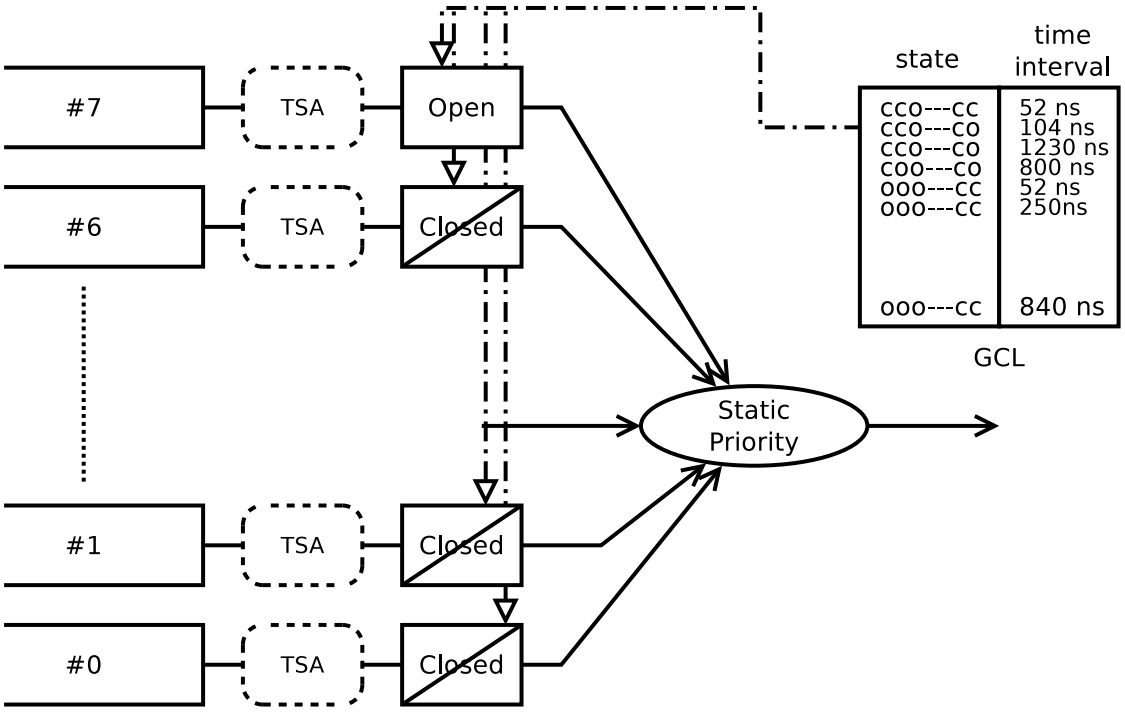

Figure 6 Architecture of a TSN output port.

that should not go this route, and then a "Flow metering" process, before the "Queuing frame" process, in charge of copying each frame in the queues of the output ports corresponding to the adequate routing (cf. [3, Figure 8-12] for an overview of the process).

The Flow metering has been enhanced first by the Per-Stream Filtering and Policing amendment [8]. PSFP mainly added a classification of flows ("Stream filtering"), and, based on this classification

- the ability to discard frames based on their size ("Maximum SDU Size Filtering"),
- a "Stream gating" capability that is a time-driven table, that allows
  - to drop frames if the gate is "closed" at this instant,
  - to set an Internal Priority Value (IPV) to the frame, that will be used to select the traffic class of the frame (overloading the frame own priority value),
- and a "Flow metering", that checks if flows respects a token bucket specification, and can mark out-of-contract frames.

The ATS amendment adds the "ATS Eligibility Time Assignment" (also named "ATS scheduler" at the end of this chain), which is in charge of computing an "eligibility time" for each ATS frame (ie. each frame that will be forwarded to a class whose TSA is ATS). The details of this computation will be presented in Section 3.4.

This eligibility time assignment is of course the core of the ATS behaviour, but the ability to set the IPV is also of importance in order to forward ATS frames to an ATS traffic class.

The behaviour of an ATS TSA (i.e. a transmission selection algorithm that implements the asynchronous traffic-shaping scheduling) consists in selecting the frame with the oldest eligibility time earlier than the current time (if any).

This means that was is depicted as a queue in the port architecture of Figure 7 is not a FIFO queue in the implementation. For example, the switch may receive at time $t_1 = 10$ a frame, compute an eligibility time $e_1 = 20$, and forward it to an ATS "queue" q at time $t'_1 = 11$. It can

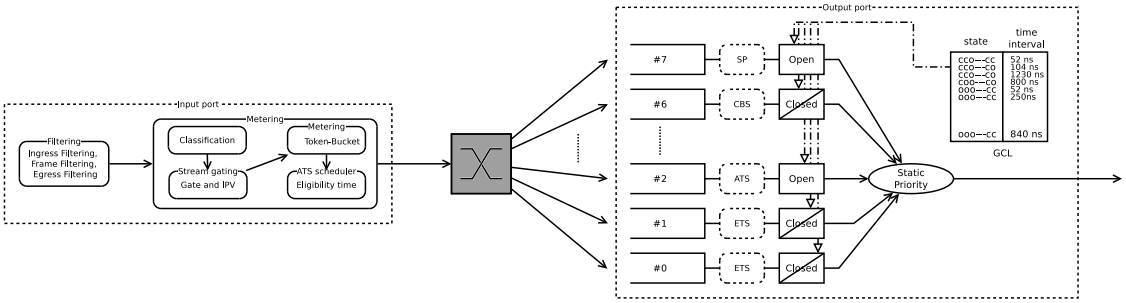

Figure 7 Illustration of the global TSN forwarding process architecture.

then receive at time $t_2 = 15$ another frame, computes an eligibility time $e_2 = 18$, and forward it to the same ATS "queue" q at time $t'_2 = 16$. If the medium becomes free at time $t = 19$, the ATS transmission selection algorithm must select for transmission the second frame, the one whose eligibility time is 18. Then, the ATS "queue" is not a FIFO queue. Then, we will use in this report the term "queuing system", whereas the standard use the term "queue".

The equivalence between UBS and ATS relies on the fact that the ATS eligibility time corresponds to the instant when the frame would have been forwarded to the ready queue in UBS.

The ATS architecture is depicted in Figure 7 (with a single input port and a single output port, for readability).

Note that this figure relies on the assumption that the switch is build as an assembly of input ports, doing metering, and output ports, doing queuing and arbitration, connected by a switching fabric. But other architectures may exist, like having a global memory for the entire switch, or having a metering hardware component shared by all input ports, etc.

In particular, Note 3 in [4, § 8.6.5.6] states that the computation of the eligibility time can be done in the input or the output port ("Whether ATS scheduler instances, ATS scheduler group instances, the scheduler instance table, and the scheduler group instance table are located in reception ports or in transmission ports is implementation specific.").

Such difference may have an impact, in particular when using both ATS and the Frame Replication and Elimination for Reliability addenda (FRER, [2]), as will be shown in Section 6.

Note that some other terms may induce confusion for a novice reader of TSN documents. A full list of abbreviations is presented in [3, § 4] but Table 1 gives a short list of some confusing ones.

3.3 ATS scheduler groups

ATS schedulers are grouped in "ATS scheduler groups". There is one group per reception port and per upstream traffic class. All ATS schedulers handling frames from the same reception port and the same upstream traffic class are in the same ATS scheduler group.

That is to say, there is one scheduler group in ATS for each shaped queue in UBS.

Each group has:

- an identifier (an integer value),
- a MaximumResidenceTime, which is a parameter that "limits the duration for which frames can reside in a Bridge" [4, § 8.6.11.3.13], that should be set at configuration,
- and a "group eligibility time", a variable shared by all ATS schedulers in the group, used in the computation of the eligibility time of each ATS frame.

Table 1 Some "confusing" acronyms in TSN.

ATS	**Asynchronous Traffic Shaping:** Mechanism defined in [4], using per flow token bucket regulation and shared queues.
TAS	**Time Aware Shaper:** Name used in the literature to reference the implementation of a Time-Triggered scheduling using [7] addendum. This acronyms is *never* used in the IEEE addenda.
TSA	**Transmission Selection Algorithm:** Generic name for the mechanisms that allows one frame in a traffic class queue to be selected for transmission. Current possible values are listed in Section 3.5. They are sometimes called "shapers".
CBS	**Credit-based Shaper:** A TSA introduced in [1], in the context of Audio-Video Bridging (AVB).
CBS	**Committed Burst Size:** Size of the burst (capacity) of the token bucket used to regulate flows in ATS.

3.4 Computation of the eligibility time

The previous sections has presented the global architecture, and this one presents the details of the ATS scheduler.

Each scheduler has two parameters: a Committed Burst Size parameter (CBS, same acronym as the Credit Based Shaper defined in [1]), and a Committed Information Rate parameter (CIR). Each scheduler is also linked to an "ATS scheduler group" (cf. Section 3.3), that has a "Maximum Residence Time" parameter, as mentioned in Section 3.3.

The main role of the scheduler is to compute the eligibility time of each frame.

To ease reproducibility of results, the algorithms are presented using the Python language, whereas obviously the standard does not impose any language.

The data structure related to the architecture is presented in Program 1 and the computation itself is in Program 2.

Consider the same example of frame arrival as in Figure 1. As expected, the Eligibility Time computed by the ATS `processFrame` algorithm is equal to the delivery time of the token bucket. The sequence of `BucketEmptyTime` is also represented in Figure 1. When the letter B in a circle stands at time 2, it means that after processing frame B, the BucketEmptyTime value is equal to 2. We have represented the sequence this way to show the graphical relation between the number of tokens and this variable: the `BucketEmptyTime` after the delivery of frame X is the intersection between the abscissa line and the line with slope r passing through the value of the bucket after the delivery of frame X (the exact relation is the core of the equivalence relation, it will be given in eq. (26)).

Also note that `BucketEmptyTime` is described as "A state variable that contains the most recent instant of time at which the token bucket of the ATS scheduler instance was empty, in seconds." [4, § 8.6.11.3.3], but such definition does not match the trace in Figure 1. For example, after handling of frame D, the `BucketEmptyTime` value is 8, whereas the token bucket was empty at instant 5 for the last time. It is also described as "the time when there are no tokens existing in the bucket" [30, § 3.3], which also does not match since they are tokens at time 8 in the trace. In the 2022 version of the standard, the description is simply "A variable that embodies the current state of the ATS scheduler instance" [5].

Program 1 Data structure (and initial values) for illustrating the computation of eligibility time in ATS.

```
class Frame:
    """A frame is simply caracterised by its length, arrivalTime time and
    eligibilityTime"""
    def __init__(self, name, arrivalTime, length):
        self.name= name # For print & debug
        self.arrivalTime= arrivalTime
        self.length= length
        self.eligibilityTime= None
```

```
class Queue:
    """A queuing system of an output port is simply represented by a list
    of Frames"""
    def __init__(self):
        self.queue= []

class ATSgroup:
    """An ATS group is linked to an output queuing system, and has a
    MaxResidenceTime parameter. It also maintains a GroupEligibilityTime
    value, shared by all ATS shedulers of the group."""
    def __init__(self, MaxResidenceTime, queue):
        self.MaxResidenceTime= MaxResidenceTime
        self.queue= queue
        # Assume init time is 0
        self.GroupEligibilityTime= 0

class ATSscheduler:
    """An ATS scheduler belongs to an ATS group. It has two parameters
    (CIR and CBS) and maintains a BucketEmptyTime value"""
    def __init__(self, CommittedInformationRate, CommittedBurstSize, atsGroup):
        self.CommittedInformationRate= CommittedInformationRate
        self.CommittedBurstSize= CommittedBurstSize
        self.group= atsGroup
        # Assume init time is 0
        self.BucketEmptyTime= - (CommittedBurstSize / CommittedInformationRate)
```

Program 2 Algorithm computing the eligibility time of an ATS frame.

```
class ATSscheduler:
```

```
    def processFrame(self, frame):
        lengthRecoveryDuration= frame.length / self.CommittedInformationRate
        emptyToFullDuration= self.CommittedBurstSize/self.CommittedInformationRate
        schedulerEligibilityTime=  self.BucketEmptyTime + lengthRecoveryDuration
        bucketFullTime= self.BucketEmptyTime + emptyToFullDuration;
        eligibilityTime =  max(frame.arrivalTime, \
                               self.group.GroupEligibilityTime, \
                               schedulerEligibilityTime)
        if (eligibilityTime <= frame.arrivalTime + self.group.MaxResidenceTime):
            self.group.GroupEligibilityTime = eligibilityTime
            if eligibilityTime < bucketFullTime:
                self.BucketEmptyTime = schedulerEligibilityTime
            else:
                self.BucketEmptyTime = schedulerEligibilityTime \
                    + eligibilityTime - bucketFullTime
            frame.eligibilityTime= eligibilityTime
            self.group.queue.queue.append( frame )
        else:
            pass # Discard invalid frame
```

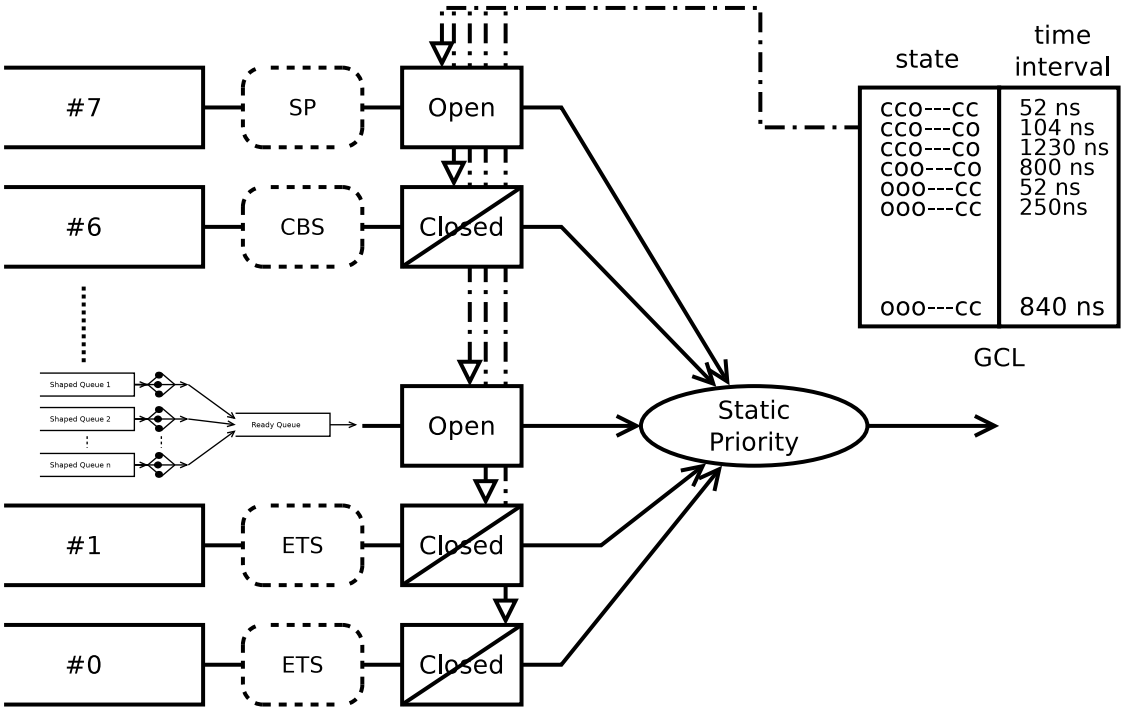

Figure 8 Architecture of a TSN output port with one UBS scheduler.

3.4.1 Tie breaker for same eligibility time

In case of frames with the same eligibility time, the queuing system must still respect some conditions already existing in previous version of the standard. The transmission order must be preserved for frames coming from the same input port and having the same "VID, priority, flow hash, destination address and source address" for unicast frames, and same "VID, priority, flow hash and destination address" for multicast frames [3, § 8.6.6].

3.4.2 Difference between ATS scheduling and ATS selection clocks

The computation of the eligibility time is done by the ATS scheduler, whereas the selection for transmission is done by the ATS transmission selection element. And both refer to a notion of current time, based on the access to a clock. In a given hardware, the ATS scheduler (can be placed in the input port) and the ATS transmission selection element (should be in the output port) may access two different clocks, having different values at the same instant. The standard formalises the differences between these clocks in [4, § 8.6.11.2], but for sake of simplicity, this report does not consider these differences. It does not consider clock drifts either. Such analysis is left for further work.

3.5 UBS as a TSN class

Whereas UBS has been initially defined for switches with only UBS scheduler, the aim was to include it into a TSN switch. This integration is straightforward and most papers focused on ATS in TSN consider implicitly a TSN switch with UBS scheduler.

For completeness, such an architecture is depicted in Figure 8.

4 State of the art

On token bucket

The idea of a data-flow regulation based on tokens in a bucket seems to appear first in [27], under the term "leaky bucket": "*Perhaps the simplest approach is the so-called 'leaky bucket' method. A counter associated with each user transmitting on a connection is incremented whenever the user sends a packet and is decremented periodically. If the counter exceeds a threshold upon being incremented, the network discards the packet. The user specifies the rate at which the counter is decremented (this determines the average bandwidth) and the value of the threshold (a measure of burstiness).*" This is a "policing" regulation, packets arriving when the counter is too small are dropped, not stored and delayed.

This leaky-bucket model is referenced by [19], but to model a traffic-shaping element, and the release of a packet decreases the number of tokens by the number of bits of the packets. Then, despite the name, it correspond to what is currently called a token bucket.

The term "token bucket filter" is introduced by [10] in its modern sense "*A token bucket filter is characterized by two parameters, a rate r and a depth b. One can think of the token bucket as filling up with tokens continuously at a rate r, with b being its maximal depth. Every time a packet is generated p tokens are removed from the bucket, where p is the size of the packet.*" It also gives the formal definition of "*the number of tokens residing in the bucket after the i'th packet leaves*" as

$$n_0 = b \qquad \forall i > 0 : n_i = \min\{b, n_{i-1} + (t_i - t_{i-1})r - p_i\} \tag{1}$$

with t_i and p_i respectively the release time and the size of the i-th packet. Up to variable names, and the delay introduced by the interleaving of flows, it corresponds to eq. (7).

A discussion between "leaky bucket" and "token bucket" can be found in [21, pp. 407 – 411].

On ATS algorithm

The initial proposition of a per-flow shaper for TSN has been done in [20], under the name Urgency Based Shaped (UBS).

An algorithm for the per-flow token bucket emulator (TBE) is given in [20, Listing 2]. Its expression is very similar to our modelling of the token-bucket-based interleaved regulator (Section 5.3), since it maintains the number of tokens during the bucket lifetime.

But the ATS standard main algorithm, the `ProcessFrame` function, given in [4, § 8.6.11.3], maintains another variable, the `BucketEmptyTime`, without any reference to the algorithm in [20, Listing 2]. It only refers to [21, pp. 407 – 411] but this reference presents the token bucket only with natural language and provides no explicit algorithm.

Then, in order to do a formal proof of equivalence, we have redone the token bucket interleaved regulator model from scratch, not by re-using [20, Listing 2].

Both the algorithms of TBE and `ProcessFrame` are presented in [30], but nothing is said on their differences and relations. This work presents ATS and a Paternoster algorithms, and provides some simulations to compare their mean delays, buffer occupancy and loss rates.

On ATS performances

Some delay bounds provided by UBS are provided in [20]. It also shows that the reshaping mechanism does not introduce any "additional delay" in the worst case [20, § IV.C], if conditions QAR1, QAR2, QAR3 are satisfied. This "reshaping for free" property is well-known "was well known for per-flow shapers and per-flow service curve elements" [14] but the significant fact is that it still hold whereas UBS uses a interleaved shaping.

The interleaved regulator has been modelled and generalised under the notion of Π-regulator in [14]. It also generalises the "reshaping-for-free" property. It states that, under reasonable routing assumptions (inspired by QAR1, QAR2, QAR3), if a flow respects a traffic shape at the network input, then the Π-regulation that re-shapes the flow in each hop does not increase the worst per hop delay.

It is shown in [24] that an interleaved regulators can not be modelled the service curve of the network calculus theory [22].

The impact of nonideal clocks has been studied in [23], but on UBS, that is to say, without considering the difference between the clock used to compute the eligibility time and the one used to release frames whose eligibility time is not less than the current time (cf. Section 3.4.2).

The reshaping of ATS can be used to cut cyclic dependencies in the network analysis: an algorithm to minimise the number of ATS queues, LCAN, has been introduced in [25].

A simplified version of ATS has been studied in [12], in the context of inconsistent CBS/CIR parameters. It considers only one ATS instance per ATS group (there is no `GroupEligibilityTime` in the ATS algorithms). This corresponds to the case where there is a single token-bucket shaper per shaped queue, there is no "interleaved shaping", and ATS becomes a "greedy shaper" [9, 13]. The criteria is the bound on worst delay computed with a network calculus model.

A new flow control strategy, *Gate Controlled Asynchronous Traffic Shaping* (GCATS) that dynamically adapts the Gate Control List based on the ATS eligibility time, is proposed in [11]. Like in [12], it considers only one ATS instance per ATS group. The Gate mechanism does not require a global clock synchronisation (whereas TAS does) and the experiments show an improvement of worst delay with regards to TAS for the flows of priority 7 and 6. The criteria is the worst delay measured by simulation.

A model of a TSN switch with 1) a high priority queue (called Control-Data Traffic, CDT) and 2) two queues of priority 6 and 5 (called queues A and B, using the AVB naming [1]) shaped by a sequence of ATS and CBS is presented in [17]. The ATS model is based on the interleaved regulator model from [14]. This proposal of sequence of ATS and CBS shapers is natural when considering the UBS architecture, where the UBS interleaved shaping is done after the shaped queues, and a CBS TSA can be introduced after the ready queue and before the gate (cf. Figure 8). But TSN allows only one TSA per class, so one can not put both an ATS and a CBS TSA on a single queue.

A comparison of the performances of various TSN transmission selection algorithm (called "TSN shapers"): TAS, CBS and ATS/UBS, and also combinations of such mechanisms can be found in [28]. This paper also investigates the combinations of different "shapers" on the same queue. In particular, like [17], it studies the performances of the combination of ATS and CBS on the same queue ("Even though ATS+CBS on the same queue is not supported by the standards, their combination is still worthwhile to be investigated from a research perspective."). The performance criteria is the bound on worst delay computed with a network calculus model.

A comparison of CBS, TAS and ATS based on simulation of an automotive case study is done in [29]. On the considered case study, ATS appears as the best trade-off.

TAS and ATS are compared in [18]. A limited version of TAS is considered with only two entries in the GCL list (i.e. two lines in the GCL illustrated in Figure 6), where a protected window of fixed size, devoted to high priority traffic, alternate with another window, with it own fixed size, devoted to best effort traffic. A mechanisms, called "Adaptive Bandwidth Sharing/Adaptive Slotted Window" (ABS/AWS), can dynamically update the sizes of the two windows. TAS, ABS/AWS and ATS are compared on a ring made of six nodes. The performance criteria are the mean and maximal packet delay measured during the simulation.

Based on the results of [14], the relations between FRER and ATS in the case on non ideal clocks has been studied in [26], under the assumption that FRER duplicates discarding is put before ATS. But our current understanding is that discarding is done after the computation of the eligibility time, as presented in Section 6.

Note that all works considering bounds on worst-case delay are based on UBS, even if they mainly use the name "ATS" (except [12] that uses ATS). The studies based on simulation use the standard ATS algorithm (with a simplification due to the single ATS instance per ATS group in [11]).

5 Equivalence between the theoretical and the standard models

To prove that UBS and ATS models are equivalent, we are going to prove that the eligibility time computed by the ATS for a frame is the same that the date when this frame would have been forwarded to the ready queue in UBS.

5.1 Partial equivalence

This is not a complete equivalence since UBS assumes a FIFO behavior of each shaped queue and of the ready queue. Then, it has to maintain the reception order between frames having the same release dates, whereas the ATS transmission selection must select the frame with the smaller eligibility time but in case of equality, its tiebreaker (presented in Section 3.4.1) allows to invert the order between two frames coming from the same input port but with different destination addresses.

Moreover, the token-bucket algorithm assigns an infinite release time to a frame whose size is larger than the burst size, whereas the ATS algorithm computes a finite eligibility time in this case.

Last, ATS associates to each group a `MaximumResidenceTime`, used to drop frames whose waiting time would be too large, whereas this does not exist for a token bucket.

5.2 Building the equivalence

The equivalence is not straightforward since the common presentation of the token bucket [20, Listing 2], [21, pp. 407 – 411] is based on a variable representing the number of tokens, whereas the algorithm in [7] maintains another variable, the `BucketEmptyTime`.

The core of the contribution relies on the exhibition of the relation between these variables, already illustrated in Figure 1.

The working plan is the following: we are going to consider one single shaped queue. Let A_n be the arrival date of the n-th message in the queue, F_n the flow it belongs to, and L_n its length. Then, we can define D_n the departure instant when it is forwarded in the ready queue, and eT_n the eligibility time computed by the ATS scheduler. The aim is to prove that $\forall n : D_n = eT_n$ (the notations are inspired from [14] to ease comparison and further works).

The two first steps consists in building the sequences D_n (Section 5.3) and eT_n (Section 5.4). The equivalence itself is given in Section 5.5.

5.3 Modelling the interleaved shaper

Let first model a token bucket, with parameter (r, b). As recalled in Section 2.1, a shaping token bucket allows a frame to be forwarded at instant t if the number of tokens at time t is not less than the frame size. So, one need to model this number. Since the bucket is replenished continuously,

Table 2 Main notations.

UBS related notations (interleaved shaping)	
A_n	Arrival date of the n-th message
F_n	Flow of the n-th message
L_n	Length of the n-th message
D_n	Departure of the n-th message
$n \ominus 1$	Index of the previous frame of the same flow, cf. eq. (2)
B_n^f	Number of tokens in the bucket devoted to f *just after* the departure of the n-frame.
ATS related notations (Program 2)	
lRD_n	value of `lengthRecoveryDuration` at n-th call
$eTFD_n$	value of `emptyToFullDuration` at n-th call
sET_n	value of `schedulerEligibilityTime` at n-th call
bFT_n	value of `bucketFullTime` at n-th call
eT_n	value of `eligibilityTime` at n-th call
GET_n	value of `GroupEligibilityTime` at n-th call
BET_n	value of `BucketEmptyTime` at n-th call

Table 3 Illustration of the $\ominus 1$ operator on a sequence of 8 frames belonging to flows 10 and 20.

i	0	1	2	3	4	5	6	7
F_i	10	10	20	20	10	10	10	20
$i \ominus 1$	0	0	0	2	1	4	5	3

we may propose to model the number of tokens as a functions of time. But when a frame is selected for transmission, it instantaneously consumes the tokens. Then, it is more convenient to build the number of tokens as a sequence, depending on arrival instants, departure instants and frames sizes.

Consider an instant t when a frame arrives at head of queue. Let B denote the number of tokens in the bucket *just after* the last departure of a frame, occurred at time D. Then, the current number of tokens is $B + r(t - D)$ and the frame can be released if $L \leq B + r(t - D)$. Otherwise, it has to wait up to time $t' = \min\{u > t \mid L \geq B + r(t - D)\}$ i.e. $t' = \frac{L-B}{r} + D$.

Now, we can define the departure sequence D_n as a function of A_n, L_n and F_n.

Note that the model presented here, including the notations (A, L, F, D), comes from [14]. The evolution rules that will be given as text in IR1-IR6 and mathematical expressions in eq. (4)–(7) are a specific case of the evolution rules given in the more generic context of [14]. All these rules are given for completeness.

Let B_n^f the number of tokens in the bucket devoted to the flow f *just after* the departure of the n-frame.

Let also introduce a notation $\ominus 1$, that given an index n returns the previous index of a packet of the same flow, defined as

$$n \ominus 1 = \sup\{n' < n \mid F_{n'} = F_n\} \tag{2}$$

with the convention that $\sup \emptyset = 0$. For example, if the queue receives two messages of flow 10, then two messages of flow 20, and one last message of flow 1, then $F_0 = F_1 = 10$, $F_2 = F_3 = 20$, $F_4 = 10$, and $0 \ominus 1 = 0$, $1 \ominus 1 = 0$, $2 \ominus 1 = 0$, $3 \ominus 1 = 2$, $4 \ominus 1 = 1$. More can be found in Table 3.

Then, the behaviour of an interleaved regulator regulating each flow f with rate r^f and burst b^f can be given using six rules, the first fours ones considering the transmission time, the last two ones managing the number of tokens in the bucket.

IR1 If the n-th frame arrives (at A_n) in an empty queue, and there are enough tokens at this instant, then it is immediately forwarded,

IR2 If the n-th frame arrives (at A_n) in an empty queue, and there are not enough tokens at this instant, then it has to wait until its bucket is replenished up to having enough tokens,

IR3 If a frame arrives in a non empty queue n-th, it will becomes the head of queue at time D_{n-1}, and if there are enough tokens at this instant, then it is forwarded at this instant,

IR4 If a frame arrives in a non empty queue n-th, it will becomes the head of queue at time D_{n-1}, and if there are not enough tokens at this instant, then it has to wait until its bucket is enough replenished,

IR5 Each buffer replenishes at its rate up to the maximal burst.

IR6 When a frame is emitted, the number of tokens is decreased by the frame size.

For the following, keep in mind that, if $r > 0$

$$L \geq B + r(t - D) \iff t \geq \frac{L - B}{r} + D. \tag{3}$$

Then the condition "there are enough tokens at time t for a frame of size L" can be expressed as $t \geq \frac{L-B}{r} + D$ if D is the last departure instant for this token bucket, and if B was the number of tokens after the departure.

Then, the behaviour of an interleaved regulator regulating each flow f with rate r^f and burst b^f can be defined as

$$\forall f : B_0^f = b^f \tag{4}$$

$$D_0 = 0 \tag{5}$$

$$\forall n > 0 : D_n = \begin{cases} A_n & \text{if } A_n \geq D_{n-1}, A_n \geq \frac{L_n - B_{\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \\ \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} & \text{if } A_n \geq D_{n-1}, A_n < \frac{L_n - B_{\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \\ D_{n-1} & \text{if } A_n < D_{n-1}, D_{n-1} \geq \frac{L_n - B_{\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \\ \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} & \text{if } A_n < D_{n-1}, D_{n-1} < \frac{L_n - B_{\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \end{cases} \tag{6}$$

$$B_n^{F_n} = \min\left\{b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_n}(D_n - D_{n\ominus 1})\right\} - L_n. \tag{7}$$

Equation (6) is a re-writing of the rules IR1-IR4 (if $A_n \geq D_{n-1}$, the n-th message arrived after the departure of the previous message, ie. it arrives in an empty queue – the "enough token" condition comes from eq. (3)). Equation (7) is based on rules IR5-IR6, but the replenishment is not given for all instants, only from departure to departure.

Now, notice that the expression "$x = u$ if $u \geq v$, v otherwise" is a definition of a maximum, so Equation (6) can be simplified into

$$D_n = \max\left\{A_n, D_{n-1}, \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1}\right\}. \tag{8}$$

The expression in eq. (8) is a specific case of the generic rule [14, eq. (44)], for a token-bucket shaper. A difference is that eq. (8) explicitly considers the number of tokens in the bucket, whereas it is implicitly defined using the full arrival history in [14, eq. (30)], leading to [14, eq. (47)]. In fact, eq. (8) is based on a practical implementation of a token bucket shaper, whereas [14, eq. (47)] is the max-plus equation, easier "to derive formal properties of such shapers" [14, § IV.C].

5.4 Modelling the eligibility time algorithm

The computation of eligibility time given in Program 2 can be also modelled as a sequence quite easily since each variable is assigned only once per call. The code related to the Maximum Residence Time will be ignored. We also assume that there is no parallelism between the schedulers of the same group and that the functions calls are made in the order of arrival, i.e. the n-th frame (with parameters A_n, L_n) is handled in the n-th call of the function.

So, to each variable will be assigned a sequence, where the n-th value corresponds to the n-th call of the function. For notation simplicity, the sequence associated to a variable is made of its first letter and the sequence of capital letters (e.g. `lengthRecoveryDuration` becomes lRD).

To make the equivalence clearer, the frame `length`, `CommittedInformationRate` and `CommittedBurstSize` will be respectively denoted L, r, b.

We also assume that there is bijection between flows and ATS scheduler instances. So, the flow name will be used as an exponent for parameters of variables related to ATS scheduler instances. For example, `self.CommittedBurstSize` becomes b^f for the instance associated to flow f. And since F_n is the flow of the n-frame, corresponding to the n-th call of the function, it becomes b^{F_n} in the sequence.

Also note that there are only two variables that are kept from one call to another: `BucketEmptyTime`, which is local to an ATS scheduler, and each instance will have its flow name as exponent and `GroupEligibilityTime` which is global to the group.

The `BucketEmptyTime` is "initialized with a time earlier than CommittedBurstSize/ CommittedInformationRate in the past, as perceived by the ATS Scheduler Clock." [4, § 8.6.11.3.3]. We assume that the ATS clock gives always positive value, and use `-CommittedBurstSize/ CommittedInformationRate` as initialisation value.

The GroupEligibilityTime "is initialized with a time earlier or equal to the current time, as perceived by the ATS scheduler clock." [4, § 8.6.11.3.10]. We use 0 as initialisation value.

Then, Program 2 can be transformed into the following sequence:

$$lRD_n \stackrel{def}{=} \frac{L_n}{r^{F_n}} \tag{9}$$

$$eTFD_n \stackrel{def}{=} \frac{b^{F_n}}{r^{F_n}} \tag{10}$$

$$sET_n \stackrel{def}{=} BET_{n-1}^{F_n} + lRD_n \tag{11}$$

$$bFT_n \stackrel{def}{=} BET_{n-1}^{F_n} + eTFD_n \tag{12}$$

$$eT_n \stackrel{def}{=} \max\{A_n, GET_{n-1}, sET_n\} \tag{13}$$

$$GET_n \stackrel{def}{=} eT_n \tag{14}$$

$$BET_n^{F_n} \stackrel{def}{=} \begin{cases} sET_n & \text{if } eT_n < bFT_n \\ sET_n + eT_n - bFT_n & \text{otherwise.} \end{cases} \tag{15}$$

Now, we can do a first simplification: each variable X^f is modified only when `self` correspond to the flow f, i.e. by a call such that $F_n = f$, and between two calls, it keeps the previous value: $\forall n, \forall m : n > m \geq n \ominus 1 \implies X_m^{F_n} = X_{n\ominus 1}^{F_n}$. In particular, $X_{n-1}^{F_n} = X_{n\ominus 1}^{F_n}$.

By applying this relation, and by replacing lRD_n, $eTFD_n$, and sET_n by their definition, the sequences can be simplified into:

$$\forall f : BET_0^f = -\frac{b^f}{r^f} \tag{16}$$

$$GET_0 = 0 \tag{17}$$

$$\forall n > 0 : bFT_n = BET_{n\ominus 1}^{F_n} + \frac{b^{F_n}}{r^{F_n}} \tag{18}$$

$$eT_n = \max\left\{A_n, eT_{n-1}, BET_{n\ominus 1}^{F_n} + \frac{L_n}{r^{F_n}}\right\} \tag{19}$$

$$GET_n = eT_n \tag{20}$$

$$BET_n^{F_n} = \begin{cases} BET_{n\ominus 1}^{F_n} + \frac{L_n}{r^{F_n}} & \text{if } eT_n < bFT_n \\ eT_n + \frac{L_n - b^{F_n}}{r^{F_n}} & \text{otherwise} \end{cases} \tag{21}$$

$$= \frac{L_n}{r^{F_n}} + \begin{cases} BET_{n\ominus 1}^{F_n} & \text{if } eT_n < BET_{n\ominus 1}^{F_n} + \frac{b^{F_n}}{r^{F_n}} \\ eT_n - \frac{b^{F_n}}{r^{F_n}} & \text{otherwise} \end{cases} \tag{22}$$

$$= \frac{L_n}{r^{F_n}} + \begin{cases} BET_{n\ominus 1}^{F_n} & \text{if } eT_n - \frac{b^{F_n}}{r^{F_n}} < BET_{n\ominus 1}^{F_n} \\ eT_n - \frac{b^{F_n}}{r^{F_n}} & \text{otherwise} \end{cases} \tag{23}$$

$$= \frac{L_n}{r^{F_n}} + \max\left\{BET_{n\ominus 1}^{F_n}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\}. \tag{24}$$

5.5 Equivalence between UBS and ATS

Here comes the main result. Keep in mind that, in this context, UBS denotes only the Token Bucket Emulation version of the Urgency-Based Scheduler, as told in the introduction.

▶ **Theorem 1** (Equivalence between UBS and ATS). *Let G be a set of flows, and for each flow $f \in G$, let $r^f, b^f \in \mathbb{R}^+$, $r^f > 0$, $b^f > 0$. Let A_n, L_n, F_n be infinite sequences with $A_n \geq 0$, $F_n \in G$, $b^{F_n} \geq L_n > 0$.*

Then, the sequences B_n, D_n defined in equations (4), (6), (7), *and the sequences eT_n, GET_n, BET_n defined in equations* (19), (20), (24) *satisfy*

$$\forall n > 0 : D_n = eT_n. \tag{25}$$

The proof relies on the relation between the `BucketEmptyTime`, the `EligibilityTime`, the rate and the bucket value, as illustrated in Figure 1 and explained in Section 3.4. The formal relation is presented in eq. (26).

The model used in this theorem assume perfect clocks, no frame loss due to buffer overflow, and does not model the `MaximumResidenceTime`.

Proof. The proof is made by induction, and will use a stronger assumption, involving not only the equality between departure dates and eligibility time but also between the bucket state and the `BucketEmptyTime`

$$\forall n > 0 : \qquad D_n = eT_n, \qquad BET_n^{F_n} = eT_n - \frac{B_n^{F_n}}{r^{F_n}}. \tag{26}$$

1. Base case: $n = 1$. Consider first the token bucket sequence.

$$D_1 = \max\left\{A_1, 0, \frac{L_n - B^{F_n}_{n\ominus 1}}{r^{F_1}} + 0\right\} = \max\left\{A_1, 0, \frac{L_n - b^{F_n}}{r^{F_1}} + 0\right\} \tag{27}$$

$$= A_1 \qquad \text{since } b^{F_n} \geq L_n \tag{28}$$

$$B^{F_1}_1 = \min\left\{b^{F_n}, B^{F_n}_{n\ominus 1} + r^{F_1}(D_1 - D_0)\right\} - L_n \tag{29}$$

$$= \min\left\{b^{F_n}, b^{F_n} + r^{F_1}(D_1 - D_0)\right\} - L_n \tag{30}$$

$$= b^{F_1} - L_1. \tag{31}$$

Now, the ATS sequence gives:

$$eT_1 = \max\left\{A_1, GET_0, BET^{F_1}_0 + \frac{L_1}{r^{F_1}}\right\} \tag{32}$$

$$\overset{(16),(17)}{=} \max\{A_1, 0, 0\} = A_1 \tag{33}$$

$$GET_1 = eT_1 = A_1 \tag{34}$$

$$BET^{F_1}_1 = \frac{L_1}{r^{F_1}} + \max\left\{BET^{F_n}_0, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{35}$$

$$= \frac{L_1}{r^{F_1}} + \max\left\{-\frac{b^{F_1}}{r^{F_1}}, A_1 - \frac{b^{F_1}}{r^{F_1}}\right\} \tag{36}$$

$$= A_1 - \frac{b^{F_1} - L_1}{r^{F_1}}. \tag{37}$$

This allow to verify the property in eq. (26) at base case $n = 1$.

$$D_1 = A_1 = eT_1 \qquad\qquad BET^{F_n}_n = A_1 - \frac{b^{F_1} - L_1}{r^{F_1}} = eT_1 - \frac{B^{F_1}_n}{r^{F_n}}.$$

2. Induction step: assume the eq. 26 holds for any index up to $n - 1$, with $n > 2$. In fact, we need to consider two cases: either $n \ominus 1 = 0$, meaning that this is the first packet of the flow F_n, or $n \ominus 1 > 0$.

2.1 Assume $n \ominus 1 = 0$. It is very similar to the base case. Consider first the token bucket.

$$D_n = \max\left\{A_n, D_{n-1}, \frac{L_n - B^{F_n}_{n\ominus 1}}{r^{F_n}} + D_{n\ominus 1}\right\} = \max\left\{A_n, D_{n-1}, \frac{L_n - b^{F_n}}{r^{F_n}} + 0\right\} \tag{38}$$

$$= \max\{A_n, D_{n-1}\} \qquad \text{since } b^{F_n} \geq L_n \tag{39}$$

$$B^{F_n}_n = \min\left\{b^{F_n}, B^{F_n}_{n\ominus 1} + r^{F_n}(D_n - D_{n\ominus 1})\right\} - L_n \tag{40}$$

$$= \min\left\{b^{F_n}, b^{F_n} + r^{F_n}(D_n - D_0)\right\} - L_n \tag{41}$$

$$= b^{F_n} - L_n. \tag{42}$$

Now turn to the ATS function.

$$eT_n = \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} \tag{43}$$

by induction, $eT_{n-1} = D_{n-1}$, and in this sub-case, $BET^{F_n}_{n\ominus 1} = BET^{F_n}_0 = -\frac{b^{F_n}}{r^{F_n}}$

$$eT_n = \max\left\{A_n, D_{n-1}, -\frac{b^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}}\right\} \tag{44}$$

$$= \max\left\{A_n, D_{n-1}\right\} \tag{45}$$

$$BET^{F_n}_n = \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{46}$$

$$= \frac{L_n}{r^{F_n}} + \max\left\{-\frac{b^{F_n}}{r^{F_n}}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{47}$$

$$= eT_n - \frac{b^{F_n} - L_n}{r^{F_n}}. \tag{48}$$

This allow to verify the property in eq. (26) in this sub-case:

$$D_n = \max\left\{A_n, D_{n-1}\right\} = eT_n \qquad BET^{F_n}_n = eT_n - \frac{b^{F_n} - L_n}{r^{F_n}} = eT_n - \frac{B^{F_n}_n}{r^{F_n}}$$

2.2 Assume $n \ominus 1 \neq 0$. Let first consider D_n and eT_n:

$$D_n = \max\left\{A_n, D_{n-1}, \frac{L_n - B^{F_n}_{n\ominus 1}}{r^{F_n}} + D_{n\ominus 1}\right\} \tag{49}$$

$$eT_n = \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\}. \tag{50}$$

The induction step states that $\forall m < n : eT_m = D_m$ and $BET^{F_m}_m = eT_m - \frac{B^{F_m}_m}{r^{F_m}}$. Now, notice that, by definition of $k \ominus 1$, for any k, $F_k = F_{k\ominus 1}$, so

$$BET^{F_n}_{n\ominus 1} = BET^{F_{n\ominus 1}}_{n\ominus 1} \tag{51}$$

$$= eT_{n\ominus 1} - \frac{B^{F_{n\ominus 1}}_{n\ominus 1}}{r^{F_{n\ominus 1}}} \qquad \text{by induction hypothesis} \tag{52}$$

$$= eT_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}}. \tag{53}$$

By substitution of this expression into eq. (50)

$$eT_n = \max\left\{A_n, eT_{n-1}, eT_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + \frac{L_n}{r^{F_n}}\right\} \tag{54}$$

$$= \max\left\{A_n, D_{n-1}, \frac{L_n - B^{F_n}_{n\ominus 1}}{r^{F_n}} + D_{n\ominus 1}\right\} = D_n. \tag{55}$$

Now that the first part of the induction relation is proved, we have to prove that $BET^{F_n}_n = eT_n - \frac{B^{F_n}_n}{r^{F_n}}$. Let start with $BET^{F_n}_n$.

$$BET^{F_n}_n \overset{(24)}{=} \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{56}$$

$$\overset{(19)}{=} \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{57}$$

$$= \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, A_n - \frac{b^{F_n}}{r^{F_n}}, eT_{n-1} - \frac{b^{F_n}}{r^{F_n}}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{58}$$

Since each frame is smaller than the burst size, $L_n \leq b^{F_n}$, so the fourth term of the max is not greater than the first, leading to

$$BET_n^{F_n} = \max\left\{BET_{n\ominus 1}^{F_n}, A_n - \frac{b^{F_n}}{r^{F_n}}, eT_{n-1} - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{59}$$

$$= \max\left\{D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}}, A_n - \frac{b^{F_n}}{r^{F_n}}, D_{n-1} - \frac{b^{F_n}}{r^{F_n}}\right\} \qquad \text{by induction.} \tag{60}$$

And now, we will reduce $eT_n - \frac{B_n^{F_n}}{r^{F_n}}$ to the same expression.

$$eT_n - \frac{B_n^{F_n}}{r^{F_n}} \overset{(7)}{=} eT_n - \frac{\min\left\{b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_n}(D_n - D_{n\ominus 1})\right\} - L_n}{r^{F_n}} \tag{61}$$

$$= \frac{L_n}{r^{F_n}} + eT_n + \max\left\{-\frac{b^{F_n}}{r^{F_n}}, -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n\right\} \tag{62}$$

$$\overset{(19)}{=} \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n \\ eT_{n-1} \\ BET_{n\ominus 1}^{F_n} + \frac{L_n}{r^{F_n}} \end{cases} + \max\begin{cases} -\frac{b^{F_n}}{r^{F_n}} \\ -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n. \end{cases} \tag{63}$$

By induction hypothesis, apply eq. (26).

$$= \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n \\ D_{n-1} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}} \end{cases} + \max\begin{cases} -\frac{b^{F_n}}{r^{F_n}} \\ -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \end{cases} \tag{64}$$

$$= \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}} \\ A_n - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \\ D_{n-1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \end{cases} \tag{65}$$

$$= \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \left(\frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}}\right) \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} - D_n + A_n \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} - D_n + D_{n-1} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} - D_n + \frac{L_n}{r^{F_n}} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \end{cases} \tag{66}$$

$$= \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \left(\frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}}\right) \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} - D_n + \max\begin{cases} A_n \\ D_{n-1} \\ \frac{L_n}{r^{F_n}} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \end{cases} \end{cases} \tag{67}$$

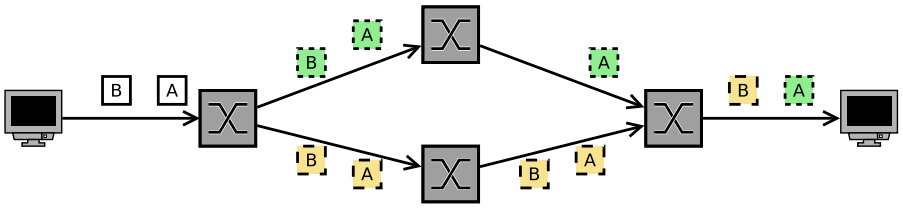

Figure 9 Illustration of FRER configuration: frames A, B are duplicated in the first switch, the B frame is lost on the upper path, and the recovery function removes one A duplicate.

Notice that $\max\left\{A_n, D_{n-1}, \frac{L_n}{r^{F_n}} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + D_{n\ominus 1}\right\}$ is equal to D_n, cf. eq (8), so

$$eT_n - \frac{B^{F_n}_n}{r^{F_n}} = \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + \left(\frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}}\right) \\ D_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + 0 \end{cases} \tag{68}$$

Since each frame is smaller than the burst size, $L_n \leq b^{F_n}$

$$eT_n - \frac{B^{F_n}_n}{r^{F_n}} = \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} \end{cases} \tag{69}$$

This ends the induction steps: from eq. (60) and (69), $BET^{F_n}_n = eT_n - \frac{B^{F_n}_n}{r^{F_n}}$. ◀

6 On ATS and FRER

Theorem 1 has proved that the eligibility times computed by ATS are the same as the release times of UBS. Nevertheless, it does not mean that one may neglect the implementation architecture. In particular, the place of the ATS scheduler in the forwarding process may interfere with other mechanisms. Here is reported an impact on Frame Replication and Elimination for Reliability addenda (FRER, [2]).

FRER allows to duplicate frames in the network, in order to improve the reliability, and to remove duplicates at joining points (as illustrated in Figure 9). We focus here on the relative position of the ATS scheduler (that computes the eligibility time) and the FRER recovery function (that remove duplicates) in the forwarding function of a switch.

If the recovery function is set *before* the ATS scheduler, the ATS scheduler will handle only the remaining frames. But since the duplicates may come from different input ports, its is not easy to implement it in input ports, since it requires to exchange some information, as illustrated in Figure 11. And if the recovery is in the output port, also is the ATS scheduler.

If the recovery function is set *after* the ATS scheduler (as in Figure 12), the ATS scheduler will compute the eligibility time of all frames, without knowing which one is going to be kept and which one is going to be discarded. Then, the frames that are kept will be delayed by discarded frames. In this case, the ATS scheduler can be put in the input ports.

Both architecture have the same functional behaviour, but the timing behaviour is not the same since both do not consume the same number of tokens.

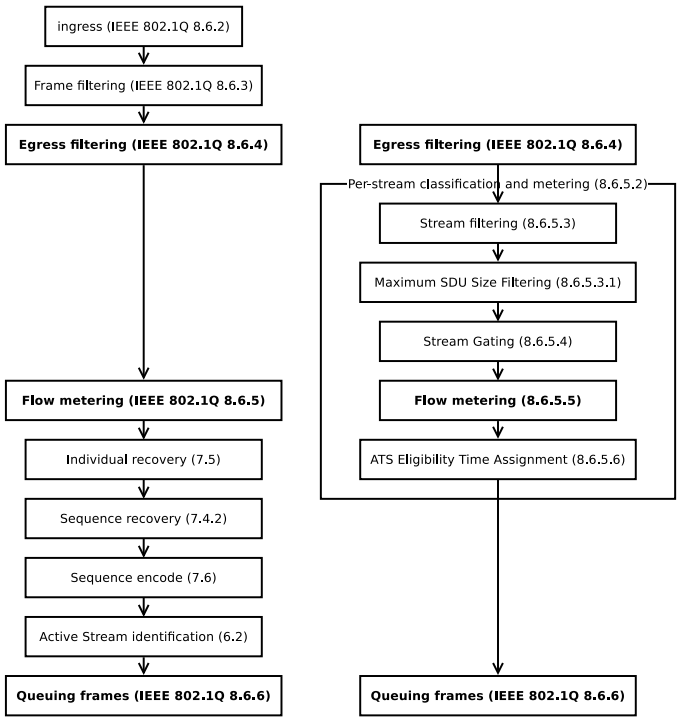

Figure 10 Inclusion in the global forwarding process of FRER (left) and ATS (right), inspired from [2, Fig. 8-2] and [4, Fig. 8-13].

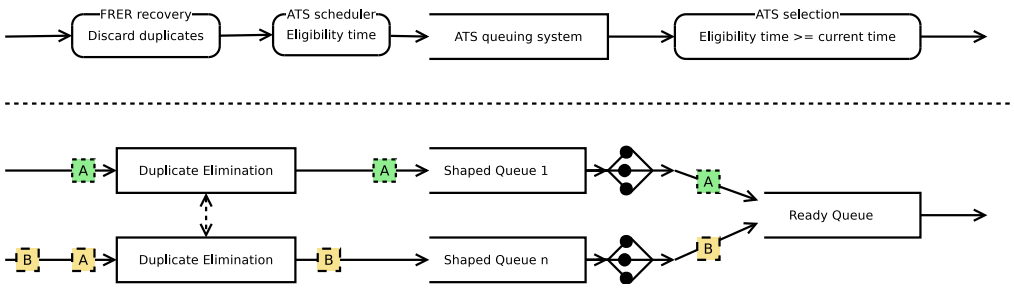

Figure 11 Setting FRER before ATS in forwarding process, and equivalent model.

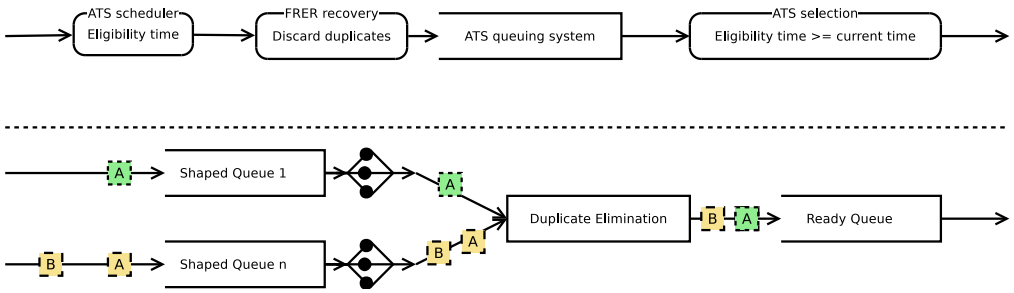

Figure 12 Setting FRER after ATS in forwarding process, and equivalent model.

Our understanding, based on the comparison of [2, Fig. 8-2] and [4, Fig. 8-13], reported side-by-side in Figure 10, is that the computation of the eligibility time is done before the discarding (§ 8.6.5.2 being a sub-part of § 8.6.5, considering that the block "Flow metering (IEEE 802.1Q 8.6.5)" is an abbreviation of the full name of paragraph 8.6.5 "Flow classification and metering"). As reported in Section 4, this would invalidates the results in [26] that does the opposite assumption.

To sum up, the ATS mechanism is implemented with three elements: the ATS scheduler, that computes eligibility time, the ATS queuing system and the ATS transmission selection, that selects in the ATS queuing system the frames whose eligibility time is less than the current time. The relative order between theses elements and the FRER recovery functions has an impact on the global sequence behaviour, matching different UBS models. Thus, the Note 3 in [4, § 8.6.5.6] that allows ATS scheduler to be implemented in input or output port has strong implications.

7 Conclusion

The Asynchronous Traffic Shaping (ATS) is one promising TSN mechanism, and a lot of research have been made to evaluate its benefits and limits. Nevertheless, most formal studies have been done on the initial proposition, UBS, whose architecture and algorithm are different from the implementation described in the ATS standard. They all were assuming that ATS is an equivalent implementation of UBS.

This paper presents both and formally proves an equivalence relation between both (the equality between the release time of frames in the theoretical model and the eligibility time computed by the standard algorithm).

This equivalence was made under the assumption that the theoretical clock and the implementation ones have the same behaviour. Since they are in fact two clocks in an ATS implementation (one for computing eligibility time at frame arrival, and one for testing eligibility time for frame departure), a further work may consider nonideal clocks, as in [23].

This paper also shows that, when ATS and the reliability mechanism FRER are both used in a switch, the order between the different elements leads to different behaviours. It may be of interest to redo the analysis of [26] when the ATS scheduler is executed before the FRER recovery.

References

1 Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams, 2010. doi:10.1109/IEEESTD.2009.5375704.

2 IEEE standard for local and metropolitan area networks – frame replication and elimination for reliability, September 2017. doi:10.1109/IEEESTD.2017.8091139.

3 IEEE standard for local and metropolitan area networks – bridges and bridged networks, 2018. doi:10.1109/IEEESTD.2018.8403927.

4 IEEE standard for local and metropolitan area networks – asynchronous traffic shaping, September 2020. doi:10.1109/IEEESTD.2020.9253013.

5 IEEE standard for local and metropolitan area networks – bridges and bridged networks, 2022.

6 IEEE standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption, 2016. doi:10.1109/IEEESTD.2016.7553415.

7 IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 25: Enhancements for scheduled traffic, 2015. doi:10.1109/IEEESTD.2016.8613095.

8 IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 28: Per-stream filtering and policing, 2017. doi:10.1109/IEEESTD.2017.8064221.

9 Cheng-Shang Chang. A filtering theory for deterministic traffic regulation. In *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, volume 2, pages 436–443 vol.2, April 1997. doi:10.1109/INFCOM.1997.644492.

10 David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):1426, October 1992. doi:10.1145/144191.144199.

11 Jiaying Feng, Qiao Li, and Bingwu Fang. A design of token bucket shaper aided with gate control list in time-sensitive networks. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pages 1–9, 2022. doi:10.1109/DASC55683.2022.9925843.

12 Hao Hu, Qiao Li, Huagang Xiong, and Bingwu Fang. The delay bound analysis based on network calculus for asynchronous traffic shaping under parameter inconsistency. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, pages 908–915, 2020. doi:10.1109/ICCT50939.2020.9295939.

13 Jean-Yves Le Boudec. Some properties of variable length packet shapers. *ACM/IEEE Transactions on Networking*, August 2002.

14 Jean-Yves Le Boudec. A theory of traffic regulators for deterministic networks with application to interleaved regulators. *IEEE-ACM Transactions On Networking*, 26(6):2721–2733, 2018. doi:10.1109/TNET.2018.2875191.

15 MEF. Subscriber ethernet service attributes. Technical Report MEF 10.4, MEF Forum, 2018. URL: http://www.mef.net/resources/technical-specifications.

16 Ehsan Mohammadpour, Eleni Stai, and Jean-Yves Le Boudec. Improved credit bounds for the credit-based shaper in time-sensitive networking. *IEEE Networking Letters*, 1(3):136–139, September 2019. doi:10.1109/LNET.2019.2925176.

17 Ehsan Mohammadpour, Eleni Stai, Maaz Mohiuddin, and Jean-Yves Le Boudec. Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping. In *30th International Teletraffic Congress (ITC 30)*, volume 02, pages 1–6, 2018. doi:10.1109/ITC30.2018.10053.

18 Ahmed Nasrallah, Akhilesh S. Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham Elbakoury. Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS). *IEEE Access*, 7:44165–44181, 2019. doi:10.1109/ACCESS.2019.2908613.

19 A. Parekh and R. Gallager. A generalised processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE transactions on networking*, June 1993. doi:10.1109/INFCOM.1992.263509.

20 Johannes Specht and Soheil Samii. Urgency-based scheduler for time-sensitive switched ethernet networks. In *Proc. of the 28th Euromicro Conference on Real-Time Systems (ECRTS 2016)*, 2016. doi:10.1109/ECRTS.2016.27.

21 A. S. Tanenbaum and D. J. Wetherall. *Computer Networks, 5th ed.* New Jersey: Prentice Hall, 2010.

22 Ludovic Thomas and Jean-Yves Le Boudec. Network-calculus service curves of the interleaved regulator, 2023. arXiv:2305.18036.

23 Ludovic Thomas and Jean-Yves Le Boudec. On time synchronization issues in time-sentive networks with regulators and nonideal clocks. *Proc. of the ACM on Measurement and Analysis of Computing Systems*, 4(27), June 2020. doi:10.1145/3392145.

24 Ludovic Thomas and Jean-Yves Le Boudec. Network-calculus service curves of the interleaved regulator. In *Proc. of the 35th international teletraffic congress (ITC 35th)*, Turin, Italy, October 2023.

25 Ludovic Thomas, Jean-Yves Le Boudec, and Ahlem Mifdaoui. On cyclic dependencies and regulators in time-sensitive networks. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 299–311. IEEE, 2019. doi:10.1109/RTSS46320.2019.00035.

26 Ludovic Thomas, Ahlem Mifdaoui, and Jean-Yves Le Boudec. Worst-case delay bounds in time-sensitive networks with packet replication and elimination. *IEEE/ACM Transactions on Networking*, pages 1–15, 2022. doi:10.1109/TNET.2022.3180763.

27 J. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 20(10), 1986. doi:10.1109/MCOM.2002.1006972.

28 Luxi Zhao, Paul Pop, and Sebastian Steinhorst. Quantitative performance comparison of various traffic shapers in time-sensitive networking. *IEEE Transactions on Network and Service Management*, 19(3):2899–2928, 2022. doi:10.1109/TNSM.2022.3180160.

29 Zifan Zhou, Juho Lee, Michael Stübert Berger, Sungkwon Park, and Ying Yan. Simulating tsn traffic scheduling and shaping for future automotive ethernet. *Journal of Communications and Networks*, 23(1):53–62, 2021. doi:10.23919/JCN.2021.000001.

30 Zifan Zhou, Michael Stübert Berger, and Ying Ruepp, Sarah Renée annd Yan. Insight into the IEEE 802.1 Qcr asynchronous traffic shaping in time sensitive network. *Advances in Science, Technology and Engineering Systems Journal*, 4(1):292–301, 2019. doi:10.25046/aj040128.

A Code usage

Some supplementary code that can be used to test the ATS `processFrame` function is given in Programs 3 and 4.

Program 3 Pretty print of Frame object.

```
class Frame:
```

```
    def __repr__(self):
        if self.eligibilityTime == None:
            return "Frame(" + self.name + "," + str(self.arrivalTime) + "," + \
                str(self.length) +")"
        else:
            return "Frame(" + self.name + "," + str(self.arrivalTime) + "," + \
                str(self.length) +")->" + str(self.eligibilityTime)
```

Program 4 Examples of tests.

```
if __name__ == "__main__":
    # Single flow in queue
    queue= Queue()
    ats_group= ATSgroup(100000, queue)
    ats_sched= ATSscheduler(1,3, ats_group)
    ats_sched.processFrame( Frame("A", 1, 2) )
    ats_sched.processFrame( Frame("B", 2, 2) )
    ats_sched.processFrame( Frame("C", 3, 3) )
    ats_sched.processFrame( Frame("D", 9, 2) )
    ats_sched.processFrame( Frame("E", 9, 2) )
    print( queue.queue )


    # ATS with two queues
    queue= Queue()
    ats_group= ATSgroup(100000, queue)
    ats_schedA= ATSscheduler(50,100, ats_group)
    ats_schedB= ATSscheduler(50,100, ats_group)

    # First frame A: delivered as soon as arrived, take all tokens
    ats_schedA.processFrame( Frame("A1", 0, 100) )
    # Second frame A, has to wait D_2 = A_1 +1 = 2
    ats_schedA.processFrame( Frame("A2", 1, 100) )
    # First frame B, enough tokens, but has to wait A2
    ats_schedB.processFrame( Frame("B1", 1, 50) )
    # Second frame B, enough tokens, delivered as soon as arrived, take all tokens
    ats_schedB.processFrame( Frame("B2", 2, 50) )
    # Third frame B, has to wait
    ats_schedB.processFrame( Frame("B3", 2, 100) )
    # Third frame A, size larger than CBS
    ats_schedA.processFrame( Frame("B3", 10, 1000) )
```