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

Marc Boyer

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

Marc Boyer
ONERA/DTIS
Marc.Boyer@onera.fr

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 Urgency Based Shaped (UBS). Several studies have exhibited properties and limitations of this solution, but they all are based on the model presented in the UBS definition [14], 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

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, be adapting or defining new arbitration policies in output ports: the credit-based shaper (CBS, [2]), the Enhancements for Scheduled Traffic (sometime called "Time Aware Shaper", [7]) and last the Asynchronous Traffic Shaping (ATS, [5]), which is the main subject of this paper.

The initial proposition was called Urgency-Based Scheduler [14]. It defines a new TSN class where flows are re-shaped after reception by a switch before being put in the output queue. This re-shaping requires specific 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.

This mechanism has been standardised as a TSN addendum, under the name "Asynchronous Traffic Shaping" (ATS, [5]), but with a quite different architecture: instead of storing frames and deciding if a frame must be forwarded to the output queue or delays when it becomes the head of queue, an algorithm computes, when the frame is received, an "eligibility time", and the frame is then forwarded to the output queue, but it will be eligible for transmission only after the eligibility time.

In this paper, we will call the initial model the "theoretical model", and the mechanism presented in the standard will be called the "standard model".

Then, a lot of studies have been done on the theoretical model of ATS [10, 16, 20, 12]. This paper proves that the standard model is, up to the reordering of frames with the same eligibility time, equivalent to the theoretical model. It also shows that some implementation variability allowed by the standard (the place of some ATS sub-component in the switch) have an impact on the interactions between ATS and the Frame Replication and Elimination for Reliability mechanism (FRER, [3]),

The paper is organised as follow. Section 2 presents ATS, starting by the theoretical

model (Section 2.1) and then its standard specification (Sectin 2.2). The equivalence is given in Section 3. Providing a formal equivalence requires first to give a formal model of the ATS theoretical model (Section 3.3) and of the standard algorithm (Section 3.4). The equivalence itself is stated and proved in Section 3.5. The impact of the relative place of ATS and FRER in the forwarding process is presented in Section 4. The relation between this work and state of the art is discussed in Section 5.

2 Presentation of ATS behavior

2.1 The theoretical model

This section presents the theoretical model of ATS, as defined in [14] (up to some renaming to ease readability). It starts with some recall on the notion of token bucket in Section 2.1.1. Then, ATS is presented as the assembling of different elements. Section 2.1.2 presents the "shaped queue", a queue that is shared by different token buckets, that has been generalised under the name "interleaved regulator". Different shared queues can be assembled into an "ATS queing system", as presented in Section 2.1.3. Last, Section 2.1.4 will show how such system was designed to be integrated into an ATS switch, and givse a list of requirements for the flow to queue allocations.

2.1.1 Recall on the token bucket

The token bucket is a well known mechanism in network [19] but since it exists several flavors, and no reference naming, here is proposed a terminology (a discussion takes place in Section 5).

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 is 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.), it is delayed up to having enough tokens. The output is then "shaped" or "regulated". It is illustrated in Figure 1.
- 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.
- classifier token bucket (cTB): In classifier, there is no queue. When a frame arrives, if there are not enough tokens, the frame is marked with a specific tag as "out of contract" with a specific tag (a drop tag, a red or orange colour [11],...), and forwarded without modification of the bucket state.

The ATS mechanism relies on traffic shaping token bucket.

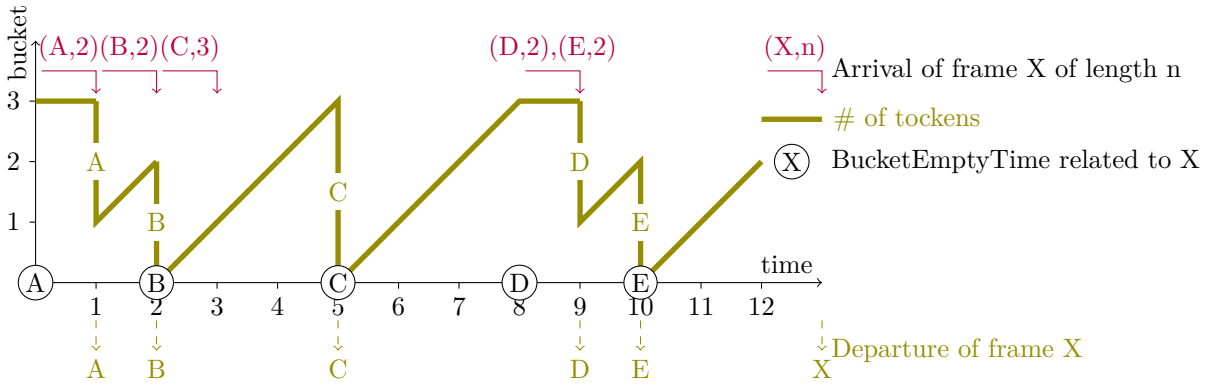

Figure 1 Illustration of behaviour of a traffic shaping token bucket with $r = 1$, $b = 3$. – The `BucketEmptyTime` will be defined in Section 2.2.3 – The bucket is initially full. When the frame A of size 2 is received at time 1, there are enough tokens, the frame is delivered and the bucket is decreased 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.

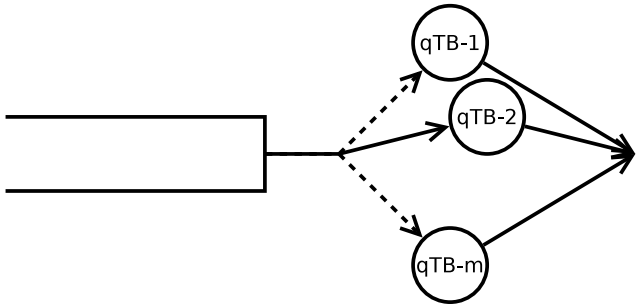

Figure 2 Shaped queue: single queue and multiple token buckets.

2.1.2 Shaped queue / interleaved regulator

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

Let first consider a sub-part, called the "shaped queue" [14, § 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, and a set of m token buckets, the i-th having parameters (r_i, b_i) (the initial paper consider two kinds of regulators, the token bucket and the Length Rate Quotient (LQR) but since the standard only consider token bucket, its is the only one presented in this paper). The specific point is that all token buckets share the same queue. This architecture in isolation is presented in Figure 3.

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 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. 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

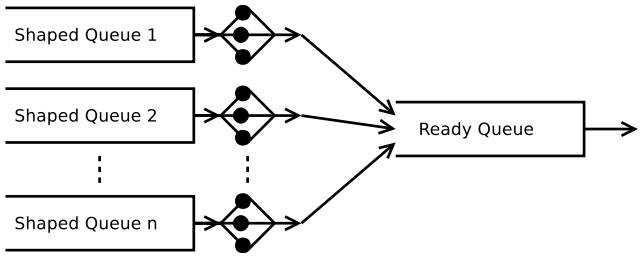

Figure 3 An ATS "queuing system": several shaped queues and one ready queue.

is delayed up to point in time when the replenishment add set enough token, 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 there 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.

Such a kind of system has been called "interleaved regulators" in [10].

2.1.3 ATS 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 flows 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 3.

Even the first presentation of UBS, in [14], suggests to have an implementation different from the theoretical model. It suggests that the implementation does not require a ready queue (and this queue is called "pseudo-queue"), 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 [14].

We found no explicit name for such a assembly, so we called it an "ATS queuing system".

2.1.4 ATS as a TSN class

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 "queuing system" (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." [5, § 8.6.6, Note 3]).

To each traffic class is associated a "Transmission selection algorithm" (TSA), that can be either "Strict priority", "Credit-based shaper", "Enhanced Transmission Selection", "ATS

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

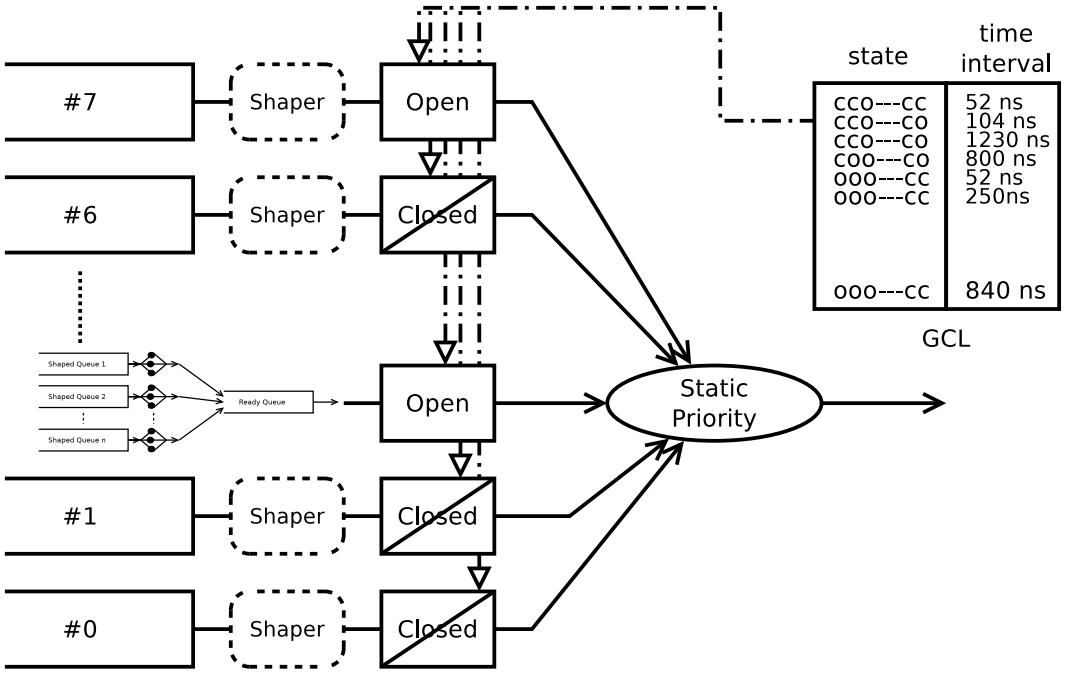

Figure 4 Architecture of a TSN output port, with one ATS queuing system (theoretical model)

Transmission Selection" or some vendor specific value [5, Table 8-6]. For the 3 first, the queuing system can be a simple FIFO queue.

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

That is to say, the head of queue of the ready queue is selected for transmission by the output port only 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 gate condition), and if there is no lower priority frame using the link (up to some fragmentation condition [6]).

Figure 4 illustrates the architecture of a TSN output port, with 8 traffic classes, without any details on the transmission selection algorithm for classes 0,1,6,7, and with an ATS queuing system in traffic class 2.

It may exist several traffic classes with an ATS transmission selection algorithm.

Then, [14] and [10] also assume that two frames with different priority levels in the previous node can not share the same shaped queue. In fact, three conditions are given in [14, § III.B]:

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

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

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

Also note that ATS (in fact, PSFP, [8]) allows to assign to a frame a priority level (the Internal Priority Level, IPV) different from the one of the priority field of the frame.

Such local assignment, with an adequate number of shaped queue (one per switch input

ATS	**Asynchronous Traffic Shaping:** Mechanism defined in [5], 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 allow one frame in a traffic class queue to be selected for transmission. Current possible values are listed in Section 2.1.4. They are often called "shapers".
CBS	**Credit-based Shaper**A TSA introduced in [2], 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.

Table 1 Some "confusing" acronyms in TSN

port sending ATS frames considered traffic class of a given port), allow to satisfy these conditions.

Note that there is no requirement that an UBS/ATS frame has the same priority in each switch: the requirement is that there is no merging of frame having different priority levels in the previous node.

2.2 The standard specification of ATS

Let first present a global view of ATS as specified in [5].

The ATS mechanism is presented and implemented in way a slightly different from the theoretical model: once an ATS frame is received, after the Egress filtering [4, § 8.6.4], it enters the Per-stream classification and metering [5, § 8.6.5.2], that does some filtering an metering (more details will be presented in Section 2.2.2). Such capabilities have been already introduced by the Per-Stream Filtering and Policing amendment, [8]). But ATS introduces new behaviour: the assignment of an "eligibility time" to each ATS frame by the "ATS scheduler"[2]. This eligibility time corresponds to the instant when the frame would have been forwarded to the ready queue in the theoretical model. Then, ATS frames are forwarded to the corresponding ATS queuing system (a traffic class). This queuing system is managed by an "ATS transmission selection algorithm". It allows a frame to be forwarded if its eligibility time is earlier than (or equal to) the current time. If several frames respect this condition, the one with the oldest eligibility time is selected. If several frames have the same eligibility time, some supplementary conditions hold, detailed in Section 2.2.3.1.

As already mentioned in Section 2.1.4, this means that the queue of a traffic class is not a single FIFO. This is why we chose in this report the term "queuing system".

This global architecture is represented in Figure 5.

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

[2] "Asynchronous Traffic Shaping (ATS) schedulers assign eligibility times to frames" [5, § 8.6.5.6].

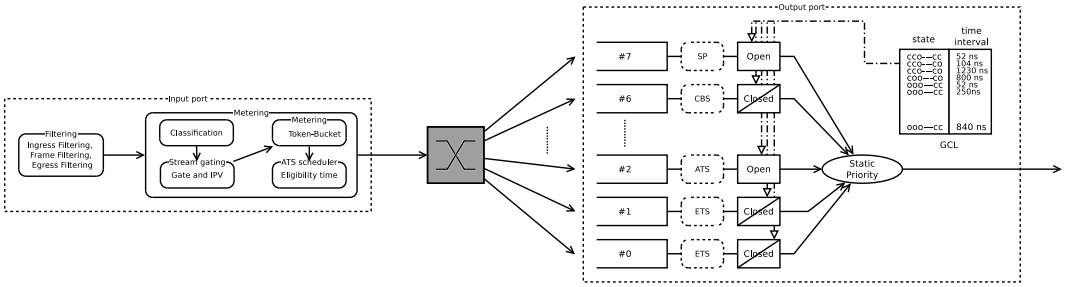

Figure 5 Illustration of the global TSN forwarding process architecture illustration

2.2.1 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 scheduler handling frames from the same reception port and the same upstream traffic class are in the same ATS scheduler group.

When considering the theoretical model, it means that where there was a shaped queue, there is one corresponding scheduler group.

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" [5, § 8.6.11.3.13],
- 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,

2.2.2 Insertion of ATS in the forwarding process

As quickly mentioned in the introduction of Section 2.2, 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 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. [4, 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), whose behaviour will be presented in Section 2.2.3.

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

This architecture is depicted in Figure 5 (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 all the switch, or having a metering hardware component shared by all input ports, etc.

In particular, Note 3 in [5, § 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, [3]), as will be shown in Section 4.

2.2.3 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 [2]), and a Committed Information Rate parameter (CIR). Each scheduler is also linked to an "ATS scheduler group" (cf. Section 2.2.1), that has a "Maximum Residence Time" parameter, as mentioned in Section 2.2.1.

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 abscisse 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, will be given in eq. (25).

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." [5, § 8.6.11.3.3], but such definition do 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" [21, § 3.3], which also does not match since they are tokens at time 8 in the trace.

2.2.3.1 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 emission order must be preserved for frames coming from the same input port and having the same "VID, priority,

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

```
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
```

Program 2 Algorithm computing the eligibility time of an ATS frame

flow hash, and destination address and source address" for unicast frames, and same "VID, priority, flow hash, and destination address" for multicast frames [4, § 8.6.6].

2.2.3.2 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 place placed in the input port) and the ATS transmission selection element (should be in the output port) may access to different clocks, having different values at the same instant. The standard formalises the differences between these clocks in [5, § 8.6.11.2], but for sake of simplicity, this report does not consider these differences. Its does neither consider clock drifts. Such analysis is left for further work.

3 Equivalence between the theoretical and the standard models

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

3.1 Partial equivalence

This is not a complete equivalence since the theoretical model assume a FIFO behavior of each shaped queue and 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 tie breaker (presented in Section 2.2.3.1) allow to invert the order of to 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.

3.2 Building the equivalence

The equivalence is not straightforward since the common presentation of the token bucket [14, Listing 2], [1, 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, 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 [10] to ease comparison and further works).

The two first steps consists in building the sequences D_n (Section 3.3) and eT_n (Section 3.4). The equivalence itself is given in Section 3.5.

3.3 Modelling the interleaved regulator

Let first model a token bucket, with parameter (r, b). As recalled in Section 2.1.1, a shaping token bucket allows a frame to be forwarded at instant t is 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, we may propose the model the number of tokens as a functions of time. But when a frame is selected for transmission, it instantaneously consume 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 $L \geq B + r(t - D)$. 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.

Let first introduce B_n^f the number of tokens in the bucket devoted 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

$$n \ominus 1 = \sup\{n' < n \mid F(n') = F(n)\} \tag{1}$$

with the convention that $\sup \emptyset = 0$.

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 emission time, the last two one on the bucket management.

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 a non empty queue, and there are not enough tokens at this instant, then it has to wait until its bucket is enough replenished,

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 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 $r \geq 0$

$$L \geq B + r(t - D) \iff t \geq \frac{L - B}{r} + D. \tag{2}$$

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

$$\forall f : B_0^f = b^f \tag{3}$$

$$D_0 = 0 \tag{4}$$

$$\forall n > 0 : D_n = \begin{cases} A_n & \text{if } A_n \geq D_{n-1}, A_n \geq \frac{L_n - B_{\ominus 1}^{F(n)}}{r^{F_n}} + D_{n \ominus 1} \\ \frac{L_n - B_{n\ominus 1}^{F(n)}}{r^{F_n}} + D_{n\ominus 1} & \text{if } A_n \geq D_{n-1}, A_n < \frac{L_n - B_{\ominus 1}^{F(n)}}{r^{F_n}} + D_{n \ominus 1} \\ D_{n-1} & \text{if } A_n < D_{n-1}, D_{n-1} \geq \frac{L_n - B_{\ominus 1}^{F(n)}}{r^{F_n}} + D_{n \ominus 1} \\ \frac{L_n - B_{n\ominus 1}^{F(n)}}{r^{F_n}} + D_{n\ominus 1} & \text{if } A_n < D_{n-1}, R_{n-1} < \frac{L_n - B_{\ominus 1}^{F(n)}}{r^{F_n}} + D_{n \ominus 1} \end{cases} \tag{5}$$

$$B_n = \min \left\{ b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_n}(D_n - D_{n\ominus 1}) \right\} - L_n \tag{6}$$

Equation 5 is a re-writing of the rules IR1-IR4, Equation 6 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 5 can be simplified into

$$D_n = \max \left\{ A_n, D_{n-1}, \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} \right\}. \tag{7}$$

3.4 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." [5, § 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." [5, § 8.6.11.3.10]. We use 0 as initialisation value.

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

$$lRD_n \stackrel{def}{=} \frac{L_n}{r^{F_n}} \tag{8}$$

$$eTFD_n \stackrel{def}{=} \frac{b^{F_n}}{r^{F_n}} \tag{9}$$

$$sET_n \stackrel{def}{=} BET^{F_n}_{n-1} + lRD_n \tag{10}$$

$$bFT_n \stackrel{def}{=} BET^{F_n}_{n-1} + eTFD_n \tag{11}$$

$$eT_n \stackrel{def}{=} \max\{A_n, GET_{n-1}, sET_n\} \tag{12}$$

$$GET_n \stackrel{def}{=} eT_n \tag{13}$$

$$BET^{F_n}_n \stackrel{def}{=} \begin{cases} sET_n & \text{if } eT_n < bFT_n \\ sET_n + eT_n - bFT_n & \text{otherwise} \end{cases} \tag{14}$$

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: $\forall n, \forall m : n > m \geq n \ominus 1 \implies X^{F_n}_{n'} = X^{F_n}_{n\ominus 1}$. In particular, $X^{F_n}_{n-1} = X^{F_n}_{n\ominus 1}$.

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

$$\forall f : BET^f_0 = -\frac{b^f}{r^f} \tag{15}$$

$$GET_0 = 0 \tag{16}$$

$$\forall n > 0 : bFT_n = BET^{F_n}_{n\ominus 1} + \frac{b^{F_n}}{r^{F_n}} \tag{17}$$

$$eT_n = \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} \tag{18}$$

$$GET_n = eT_n \tag{19}$$

$$BET^{F_n}_n = \begin{cases} BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}} & \text{if } eT_n < bFT_n \\ eT_n + \frac{L_n - b^{F_n}}{r^{F_n}} & \text{otherwise} \end{cases} \tag{20}$$

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

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

$$= \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{23}$$

3.5 Equivalence between the theoretical model and the implementation model

▶ **Theorem 1** (Equivalence between the theoretical model and the implementation model)**.** *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 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 3, 5, 6, and the sequences eT_n, GET_n, BET_n defined in equations (18), (19), (23) *satisfy*

$$\forall n > 0 : D_n = eT_n. \tag{24}$$

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 2.2.3. The formal relation is presented in eq. (25).

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`

$$\forall n > 0 : \qquad D_n = eT_n, \qquad BET_n^{F_n} = eT_n - \frac{B_n^{F_n}}{r^{F_n}} \tag{25}$$

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

$$D_1 = \max\left\{A_1, 0, \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_1}} + 0\right\} = \max\left\{A_1, 0, \frac{L_n - b^{F_n}}{r^{F_1}} + 0\right\} \tag{26}$$

$$= A_1 \qquad \text{since } b^{F_n} \geq L_n \tag{27}$$

$$B_1^{F_1} = \min\left\{b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_1}(D_1 - D_0)\right\} - L_n \tag{28}$$

$$= \min\left\{b^{F_n}, b^{F_n} + r^{F_1}(D_1 - D_0)\right\} - L_n \tag{29}$$

$$= b^{F_1} - L_1 \tag{30}$$

Now, the ATS sequence gives:

$$eT_1 = \max\left\{A_1, GET_0, BET_0^{F_1} + \frac{L_1}{r^{F_1}}\right\} \tag{31}$$

$$= \max\{A_1, 0, 0\} = A_1 \tag{32}$$

$$GET_1 = eT_1 = A_1 \tag{33}$$

$$BET_1^{F_1} = \frac{L_1}{r^{F_1}} + \max\left\{BET_0^{F_n}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{34}$$

$$= \frac{L_1}{r^{F_1}} + \max\left\{-\frac{b^{F_1}}{r^{F_1}}, A_1 - \frac{b^{F_1}}{r^{F_1}}\right\} \tag{35}$$

$$= A_1 - \frac{b^{F_1} - L_1}{r^{F_1}} \tag{36}$$

This allow to verify the property in eq. (25) at base case $n = 1$.

$$D_1 = A_1 = eT_1 \qquad BET_n^{F_n} = A_1 - \frac{b^{F_1} - L_1}{r^{F_1}} = eT_1 - \frac{B_n^{F_1}}{r^{F_n}}$$

2. Induction step: assume the eq. (25) **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.

$$D_n = \max\left\{A_n, D_{n-1}, \frac{L_n - B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1}\right\} = \max\left\{A_n, D_{n-1}, \frac{L_n - b^{F_n}}{r^{F_n}} + 0\right\} \tag{37}$$

$$= \max\{A_n, D_{n-1}\} \qquad \text{since } b^{F_n} \geq L_n \tag{38}$$

$$B_n^{F_n} = \min\left\{b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_n}(D_n - D_{n\ominus 1})\right\} - L_n \tag{39}$$

$$= \min\left\{b^{F_n}, b^{F_n} + r^{F_n}(D_n - D_0)\right\} - L_n \tag{40}$$

$$= b^{F_n} - L_n \tag{41}$$

Now turn to the ATS function.

$$eT_n = \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} \tag{42}$$

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

$$= \max\left\{A_n, D_{n-1}, -\frac{b^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}}\right\} \tag{43}$$

$$= \max\left\{A_n, D_{n-1}\right\} \tag{44}$$

$$BET^{F_n}_n = \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{45}$$

$$= \frac{L_n}{r^{F_n}} + \max\left\{-\frac{b^{F_n}}{r^{F_n}}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{46}$$

$$= eT_n - \frac{b^{F_n} - L_n}{r^{F_n}} \tag{47}$$

This allow to verify the property in eq. (25) 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.

$$D_n = \max\left\{A_n, D_{n-1}, \frac{L_n - B^{F_n}_{n\ominus 1}}{r^{F_n}} + D_{n\ominus 1}\right\} \tag{48}$$

$$eT_n = \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} \tag{49}$$

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

$$BET^{F_n}_{n\ominus 1} = BET^{F_{n\ominus 1}}_{n\ominus 1} \tag{50}$$

$$= eT_{n\ominus 1} - \frac{B^{F_{n\ominus 1}}_{n\ominus 1}}{r^{F_{n\ominus 1}}} \qquad \text{by induction hypothesis} \tag{51}$$

$$= eT_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} \tag{52}$$

By substitution of this expression into eq. (49)

$$eT_n = \max\left\{A_n, eT_{n-1}, eT_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + \frac{L_n}{r^{F_n}}\right\} \tag{53}$$

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

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 $BET^{F_n}_n$.

$$BET^{F_n}_n \overset{(23)}{=} \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, eT_n - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{55}$$

$$\overset{(18)}{=} \frac{L_n}{r^{F_n}} + \max\left\{BET^{F_n}_{n\ominus 1}, \max\left\{A_n, eT_{n-1}, BET^{F_n}_{n\ominus 1} + \frac{L_n}{r^{F_n}}\right\} - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{56}$$

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

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

$$BET_n^{F_n} = \max\left\{BET_{n\ominus 1}^{F_n}, A_n - \frac{b^{F_n}}{r^{F_n}}, eT_{n-1} - \frac{b^{F_n}}{r^{F_n}}\right\} \tag{58}$$

$$= \max\left\{D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}}, A_n - \frac{b^{F_n}}{r^{F_n}}, D_{n-1} - \frac{b^{F_n}}{r^{F_n}}\right\} \qquad \text{by induction} \tag{59}$$

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

$$eT_n - \frac{B_n^{F_n}}{r^{F_n}} \overset{(6)}{=} eT_n - \frac{\min\left\{b^{F_n}, B_{n\ominus 1}^{F_n} + r^{F_n}(D_n - D_{n\ominus 1})\right\} - L_n}{r^{F_n}} \tag{60}$$

$$= \frac{L_n}{r^{F_n}} + eT_n + \max\left\{-\frac{b^{F_n}}{r^{F_n}}, -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n\right\} \tag{61}$$

$$\overset{(18)}{=} \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n \\ eT_{n-1} \\ BET_{n\ominus 1}^{F_n} + \frac{L_n}{r^{F_n}} \end{cases} + \max\begin{cases} -\frac{b^{F_n}}{r^{F_n}} \\ -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \end{cases} \tag{62}$$

By induction hypothesis, apply eq. (25).

$$= \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n \\ D_{n-1} \\ D_{n\ominus 1} - \frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + \frac{L_n}{r^{F_n}} \end{cases} + \max\begin{cases} -\frac{b^{F_n}}{r^{F_n}} \\ -\frac{B_{n\ominus 1}^{F_n}}{r^{F_n}} + D_{n\ominus 1} - D_n \end{cases} \tag{63}$$

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

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

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

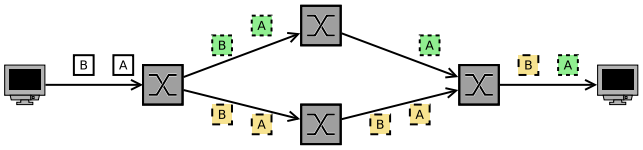

Figure 6 Illustration of FRER configuration: frames A, B are duplicated in the first switch, the B frame is lost on the upper path, and the recovery function removes one A duplicate.

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 (7), so

$$eT_n - \frac{B^{F_n}_n}{r^{F_n}} = \frac{L_n}{r^{F_n}} + \max\begin{cases} A_n - \frac{b^{F_n}}{r^{F_n}} \\ D_{n-1} - \frac{b^{F_n}}{r^{F_n}} \\ D_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + \left(\frac{L_n}{r^{F_n}} - \frac{b^{F_n}}{r^{F_n}}\right) \\ D_{n\ominus 1} - \frac{B^{F_n}_{n\ominus 1}}{r^{F_n}} + 0 \end{cases} \tag{67}$$

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

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

This ends the induction steps: from eq. (59) and (68), $BET^{F_n}_n = eT_n - \frac{B^{F_n}_n}{r^{F_n}}$. ◀

4 On ATS and FRER

Theorem 1 have proven that the eligibility time computed by ATS are the same than the release time of the theoretical model. 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 of Frame Replication and Elimination for Reliability addenda (FRER, [3]).

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 6). 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 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 require to exchange some information, as illustrated in Figure 9. And if the recovery is in the output port, also is the ATS scheduler.

If the recovery function is set *after* the ATS scheduler, 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.

To sum up, the ATS mechanism is implemented with three elements: the ATS scheduler, that computes eligibility time, and the ATS queuing system and the ATS transmission selection, that selects in the ATS queuing system the frames whose eligibility time is not less

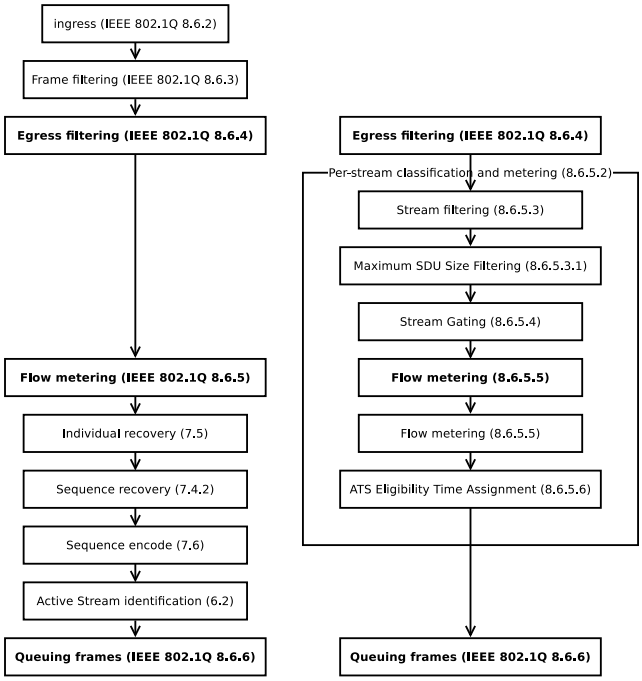

Figure 7 Inclusion in the global forwarding process of FRER (right) and ATS (left), inspired from [3, Fig. 8-2] and [**?**, Fig. 8-13]

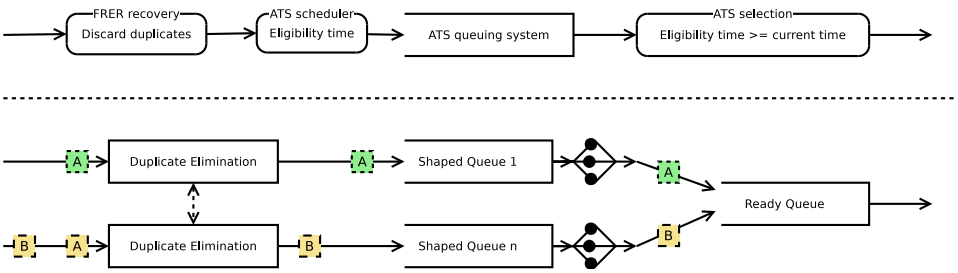

Figure 8 Setting FRER before ATS in forwarding process, and equivalent model.

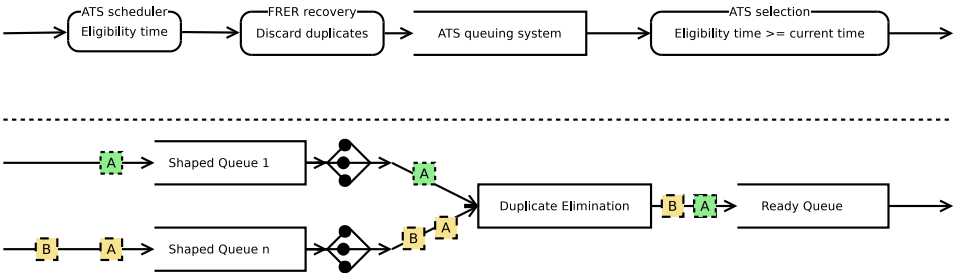

Figure 9 Setting FRER after ATS in forwarding process, and equivalent model.

than the current time. The relative order between theses elements and the FRER recovery functions have an impact on the global sequence behaviour, matching different theoretical models. Thus, the Note 3 in [5, § 8.6.5.6] that allows ATS scheduler to be implemented in input or output port have strong implications.

5 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 [19], 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 [13], 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 [9] 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

$$n_0 = b \qquad \forall i > 0 : n_i = \min\{b, n_{i-1} + (t_i - t_{i-1})r - p_i\} \tag{69}$$

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. (6).

A discussion between "leaky bucket" and "token bucket" can be found in [1, pp. 407 – 411].

On ATS algorithm

The initial proposition of a per-flow shaper fro ATS has been done in [14], under the name Urgency Based Shaped (UBS).

An algorithm for the per flow token bucket emulator (TBE) is given in [14, Listing 2]. Its expression is very similar to our modelling of the token bucket based interleaved regulator (Section 3.3), since it maintains the number of tokens during the bucket lifetime.

But the ATS standard main algorithm, the `ProcessFrame` function, given in [5, § 8.6.11.3], maintains another variable, the `BucketEmptyTime`, without any reference to the algorithm in [14, Listing 2]. It only refers to [1, 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 [14, Listing 2].

Both the algorithms of TBE and `ProcessFrame` are presented in [21], 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 perfomances

Some delay bounds provided by ATS are provided in [14].

The interleaved regulator has been modelled and generalised under the notion of Π-regulator in [10]. It also proves a very important properties of these regulator, called "reshaping-for- free". It states that, under reasonable routing assumptions (inspired by QAR1, QAR2, QAR3), if a flow respects a traffic shape at network input, then the Π-regulation that re-shapes the flow in each hop does not increase the worst per hop delay.

The impact of nonideal clocks on ATS has been studied in [15], but on the theoretical model, 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 2.2.3.2).

The reshaping of ATS can be used to cut cyclic dependencies in the network analysis: an algorithms to minimise the number of ATS queues, LCAN, has been introduced in [17].

Based on the results of [10], the relations between FRER and ATS in the case on non ideal clocks has been studied in [18], under the assumption that FRER duplicates discarding is put before ATS.

Note that all these works are based on the theoretical model of ATS.

6 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, all studies have been done on a theoretical model, whose architecture and algorithms are different from the implementation described in the standard.

This paper presents both and formally proves an equivalence relations 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 [15].

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 [18] when the ATS scheduler is executed before the FRER recovery.

References

1 *Computer Networks, 5th ed.* New Jersey: Prentice Hall, 2010.

2 Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. Technical Report IEEE 802.1Qav, IEEE, 2010.

3 Ieee standard for local and metropolitan area networks – frame replication and elimination for reliability. Technical Report 802.1CB, IEEE, September 2017.

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

5 IEEE standard for local and metropolitan area networks – asynchronous traffic shaping. Technical Report 802.1Qcr, IEEE, September 2020.

6 Ieee standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption. IEEE Standard 802.1Qbu, 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. IEEE Standard 802.1Qbv, IEEE, 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. Standard 802.1Qci, IEEE, 2017. doi:10.1109/IEEESTD.2017.8064221.

9 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, oct 1992. doi:10.1145/144191.144199.

10 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.

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

12 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, Sep. 2019. doi:10.1109/LNET.2019.2925176.

13 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.

14 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.

15 Ludovic Thomas and Jean-Yves Le Boudec. On time synchronization issues in time-sensitive networks with regulators and nonideal clocks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–41, 2020.

16 Ludovic Thomas and Jean-Yves Le Boudec. On time synchronization issues in time-sentive networks with regulators and nonideal clocks. *Proceedings of the ACM on Measurement and Analysis of Computing SystemsJune 2020 Article No.: 27*, 4(2), June 2020. URL: https://dl.acm.org/doi/10.1145/3392145, doi:10.1145/3392145.

17 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.

18 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.

19 J. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 20(10), 1986.

20 Luxi Zhao, Paul Pop, and Sebastian Steinhorst. Quantitative performance comparison of various traffic shapers in time-sensitive networking, 2021. URL: https://arxiv.org/abs/2103.13424v1, arXiv:2103.13424.

21 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.

A Code usage

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

```
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 3 Pretty print of Frame object.

```
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) )
```

Program 4 Examples of tests.