Flow Prioritization for TSN Asynchronous Traffic Shapers

Julia Caleya-Sanchez*, Jonathan Prados-Garzon*, Lorena Chinchilla-Romero*, Pablo Muñoz *, and Pablo Ameigeiras*
*Department of Signal Theory, Telematics and Communications (DTSTC), University of Granada, Spain
Emails: jcaleyas@ugr.es, jpg@ugr.es, lorenachinchilla@ugr.es, pabloml@ugr.es, pameigeiras@ugr.es

***Abstract*—Time-Sensitive Networking (TSN) technology is key to the development of current networks due to its capacity to provide a deterministic Quality of Service (QoS) mainly in terms of delay for different industrial traffic. It also simplifies management and improves the scalability of industrial networks. This articles focuses on Asynchronous Traffic Shape (ATS) for TSN. The key aspect of ATS is that by prioritizing flows delay requirement can be satisfied for each priority level. To that end, we formally formulate the problem of flow priority assignment in an network and we demonstrate the optimality of our proposed algorithm. We have compared our algorithm with the brute force search obtaining that the execution time of brute force is much higher than ours with exactly the same prioritization results.**

I. Introduction

Time-Sensitive Networking (TSN) is a set of layer 2 standards that are specified as a series of amendments to the IEEE 802.1Q standard. These standards solve critical challenges in various sectors by ensuring the deterministic transmission of flows with Quality of Service (QoS) in terms of strict requirements for latency, jitter, reliability and packet loss. Thanks to these capabilities, TSN technology is currently key to the development of deterministic networks, such as industrial or 5G networks.

TSN guarantees deterministic traffic transmission by the use of sophisticated and complex schedulers for the transmission of frames on the output ports of a TSN bridge. We can distinguish two types of schedulers defined in TSN standards: Asynchronous Traffic Shaper (ATS) and Time-Aware Shaper (TAS). In this case, we focus on asynchronous TSN networks, where a common and precise time reference is not necessary. The asynchronous TSN network uses the ATS, defined in IEEE 802.1Qcr. The ATS is based on the Urgency-Based Scheduler (UBS) proposed by Specht and Samii [1], which uses interleaved shaped queues to regulate traffic and a strict priority queue for traffic prioritization. In addition, ATS-based TSNs are more suitable for large-scale scenarios. Specifically, an ATS TSN is considered at each output port of the TSN bridge.

There are different alternatives that address the configuration of ATS-based TSN networks [1]–[3]. However, all of them present scalability problems, and it is necessary to find solutions that can handle and adapt correctly to the increase in traffic smoothly and without losing the QoS offered. Therefore, scalability is a key factor in order to be able to adapt correctly to the growing evolution of new services with more stringent requirements in terms of QoS. Additionally, due to the nature of the proposed solutions, these solutions are all computationally complex to implement.

In this work, a solution is proposed to solve the above problems, by a novel ATS-based TSN network configuration method, which minimizes the number of priority levels with respect to known techniques, under deterministic QoS requirements. The proposed solution attempts to find a feasible prioritization of a set of traffic flows in a single concrete ATS instance while fulfilling the delay requirements of the flows. This solution scales well with the increase in flows to be accommodated. That is, for a large number of flows, the proposed solution attempts to determine the feasible prioritization in an efficient way. The solution assumes that the requirements of the flows are previously known, being suitable, for example, for industrial networks where the types of traffic are known a priori. Additionally, the algorithm that develops the proposed solution presents a feasible and uncomplicated implementation and is valid for both online and offline solutions.

For evaluation purposes, we consider an industrial scenario with different types of traffic with different QoS requirements. An analysis of the degree of optimality and accuracy of the proposed algorithm is provided. Specifically, we demonstrate that proposed algorithm minimizes the number of priority levels required in an ATS instance, fulfilling the queuing delay requisites of the flows traversing the ATS instance. Furthermore, we compared the performance and flow prioritization by our algorithm with the exhaustive search of all possible configurations (brute force). The results show that our algorithm scales correctly, with execution times six orders of magnitude lower than brute force and with exactly equal prioritization results for both approaches.

The remainder of the paper is organized as follows: Section II review of the ATS description and existing work addressing the performance of an ATS-based network. Section III describes the system model and the prioritization problem and its formulation. Section IV defines the developed algorithm with its analysis and design principles. Section V provides the experimental results and section VI draws the conclusions.

This work has been partially funded by the H2020 research and innovation project 5G-CLARITY (Grant No. 871428), 6G-CHRONOS (TSI-063000-2021-28) and the Spanish Ministry of Universities (FPU Grant 21/04225).
ISBN 978-3-903176-57-7

II. Background

A. ATS Description

The ATS defines an asynchronous method for handling frames on the TSN bridge output ports [4], [5]. The TSN standards [5], which specify the ATS, are based on the UBS proposed by Specht and Samii in [6]. In [6] is considered a leaky bucket in the asynchronous shapers for flow traffic regulation. The ATS can be a practical implementation of the UBS in 802.1Q standards [5]. In this work, we adopt the nomenclature used in [6].

The queuing model of the ATS is shown in Fig. 1 [3]. For simplicity, only one egress port is shown in Fig. 1, but there is an ATS instance for each egress port of the bridge. The ATS consists of two queuing stages: i) a set of shaped queues, which are First In, First Out (FIFO) queues with an interleaved regulator, for interleaved shaping and ii) a set of priority queues. In the first stage, interleaved shaping, to perform traffic control of a set of flows, each with its own requirements, the use of a single queue (shaped queue) can be employed. The use of shaped queues before strict priority queues avoids arbitrarily large worst-case delays, because the burstiness of the flows remains constant with each hop. However, it can lead to packet losses of flows. Remarkably, in [7] LeBoundec demonstrated that the Worst-Case Queuing Delay (WCQD) is not enhanced by the use of shaped queues in the ATS. That is, placing a minimal interleaved regulator after an arbitrary FIFO queue has no negative effect on the delay for the worst case combination. The second stage uses First Come, First Served (FCFS) queues with a strict priority transmission selection algorithm. In each of the queues, all the shaped queues with the same priority level are merged.

B. Related Works

This section overviews existing works related to the solutions proposed for the flow prioritization in asynchronous TSN networks [1]–[3], [8], [9].

In [1], Specht and Samii consider a Satisfiability Modulo Theories (SMT) solver to find a feasible configurations in ATS-based networks. They propose a Topology Rank Solver (TRS) heuristic to cope with the high complexity of the pure SMT solution. Nonetheless, TRS relies on SMT for flow prioritization in at least a single ATS instance. In [3], Prados *et al.* propose a solution combining heuristic and convex optimization to seek a long-term configuration of ATS-based TSN network. Specifically, the work in [3] addresses the problem formulated in [9] which aims to minimize the probability of flow rejection. Although the cited works are armed with heuristics methods to cope with computational complexity, they uses exact optimization method to address the flow prioritization in the ATS instance, which limits their scalability with the number of flows.

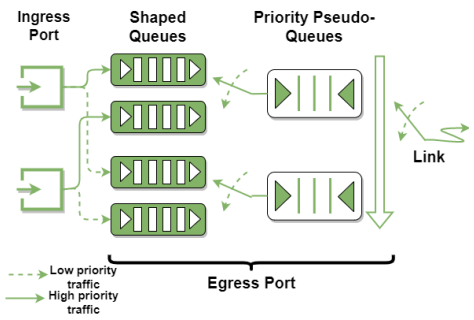

Fig. 1. ATS queuing model.

In [2] and [8], Prados *et al.* suggest an online approach based on Deep Reinforcement Learning (DRL) to determine the configuration of each flow as it arrives at the network. The requirements of the flows in this solutions are unknown and present a low capability, i.e., they depend on the network topology and have to be trained specifically for each scenario which leads to a large training time. Additionally, these works do not include a model of the flow allocation problem in asynchronous TSN networks. Moreover, all the solutions proposed are complex to implement.

In this work, unlike [2], [3], [8], [9], the proposed solution considers known flow characteristics and requirements, which is the common situation in many scenarios such as industrial networks. Furthermore, unlike the exact optimization methods considered in [1], [3], it is scalable as it allows to increase the number of flows in the scenario proving that the computational time complexity is maintained while providing a feasible prioritization. Last, remarkably, the proposed algorithm is easier to implement.

III. System Model and Problem Formulation

In this section, first we describe the considered system model. and then we formally formulate the ATS flow prioritization problem addressed in this work.

A. System Model

Let us consider an asynchronous TSN network comprising a set of ATS-based TSN bridges. There are a set of delay-sensitive flows to be conveyed through the network whose traffic is constrained by $r \cdot t + b$ [10], where r and b are the committed data rate and burst size (burstiness), respectively. The number of flows, their respective traffic features r and b, and their end-to-end (E2E) delay requisite are known beforehand. This is the common situation in industrial networks. For instance, we might have critical flows to communicate alarm events, control the motion of the operational technology devices, and steer the mobile robots through the factory floor. Each flow follows a specific path in the TSN network and its E2E delay requisite is somehow distributed among the different hops of the path.

Each TSN bridge includes an ATS instance at every egress port to handle the packets transmission at the link according to the operation described in the previous section. The ATS instances include P priority levels and enough shaped queues as to use all the priority levels regardless of the asynchronous TSN network configuration (e.g., number of ingress ports at the respective TSN bridge and prioritization considered in the

previous hop). Without loss of generality, we assume that each priority level is associated with an integer index p and lower indexes mean higher priority levels. In this way, priority 1 is the level with the highest priority. The priority levels 1 to $P-1$ are reserved to accommodate delay-sensitive traffic, whereas the priority level P is destined for best-effort traffic (e.g., remote access and maintenance in manufacturing [11]).

Given the per-flow chosen paths and delay requisite distribution among hops, there is a set of delay-sensitive flows $\mathcal{F}$ to be prioritized at each ATS instance. We assume there is a worst-case delay requisite, denoted as D_f, for each flow $f \in \mathcal{F}$ at the respective ATS instance. Thus, we can define the WCQD requisite at the ATS instance for flow f as $R_f = D_f - l_f/C$, where l_f is the maximum frame size of the flow f and C is the nominal capacity of the link handled by the ATS instance. Without loss of generality, we assume that each flow $f \in \mathcal{F}$ is associated with an integer index i according to its WCQD requisite R_f, also denoted as R_i, $R_i = R_f$. Specifically, lower indexes mean more stringent WCQD constraints, i.e., $R_{i-1} \leq R_i \leq R_{i+1}$ being the flows with indexes 1 and $F = |\mathcal{F}|$ those with the most stringent and most lenient WCQD requisites, respectively.

Let $\mathcal{F}_p$ be the set of flows allocated to a priority level p. The WCQD Q_p experienced by every flow allocated to a priority level p is upper bounded as follows [6], [7], [12]:

$$Q_p = \frac{\sum_{\forall f \in \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_p} b_f + \max_{\forall f \in \mathcal{F}_{p+1} \cup \ldots \cup \mathcal{F}_8} l_f}{C - \sum_{\forall f \in \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_{p-1}} r_f} \quad (1)$$

where r_f and b_f are the committed data rate and committed burst size (burstiness) for the flow f, respectively. Please find in Table I the primary notation considered in this work.

B. Problem Statement and Formulation

The problem addressed in this work consists in finding a feasible or satisfiable prioritization for $\mathcal{F}$ at the respective ATS instance, i.e., the delay requisites $\forall\, f \in \mathcal{F}$ are met, to minimize the number of priority levels used in it. Below is the formal formulation of the stated problem:

$$\begin{aligned} minimize \Bigg\{ & \max_{\forall f \in \mathcal{F}} P_f : \; P_f \in [1, 1-P] \cap \mathbb{N} \; \forall f \in \mathcal{F} \, (C1); \\ & Q_p \leq R_f \; \forall \, f \in \mathcal{F}_p, \, p \in [1, P-1] \, (C2); \\ & \sum_{\forall \, f \in \mathcal{F}} r_f \leq C \, (C3). \Bigg\} \end{aligned} \quad (2)$$

where $\mathbb{N}$ is the set of natural numbers. The decision variables are P_f which denotes the priority level assigned to flow $f \in \mathcal{F}$. This variables are integer and take values in the available priority levels for delay-sensitive traffic in the corresponding ATS instance, as specified in constraint *C1*.

The objective of the problem above is to minimize the required number of priority levels in the ATS instance. The motivation of choosing this optimization goal is because the cost of the asynchronous TSN network directly depends on the available priority levels in the ATS instances. The higher the number of available priorities is, the higher the deployment costs (capital expenditures) as the ATS-based TSN bridge's price raises. Moreover, it is easier to configure and operate an asynchronous TSN network whose ATS instances have a lower number of priority levels.

TABLE I
PRIMARY NOTATION

Notation	Description
$\mathcal{F}$	Set of flows to be prioritized in the ATS instance.
$\mathcal{F}_p$	Set including all the flows currently allocated to priority level p in the target ATS instance.
C	Nominal link capacity of the target ATS instance.
r_f, b_f, and l_f	Committed data rate, committed burst size, and maximum frame size of the flow f, respectively.
R_f and D_f	WCQD and delay requisites for the flow f at the target ATS instance, being $R_f = D_f - l_f/C$.
Q_p and Q_f	WCQD of priority level p and experienced by flow f.
R_1 and R_F	The most stringent and most lenient WCQD requisites among all the flows $\mathcal{F}$.
P	Maximum number of priority levels (queues) available in the target ATS instance.

Regarding the primary constraints, we must ensure that the aggregated rate traversing the ATS instance is lower than the nominal capacity (*C3*). In fact, this technological constraint is a primary assumption to derive (1) [6], [7]. On the other side, the WCQD requisites for all the flows has to be met (*C2*).

IV. ATS Flow Prioritization Algorithm

A. Design Principles

Let us start introducing some relevant propositions that can be directly proven from (1) and are behind the rationale of the proposed algorithm. Also, these propositions are cornerstone for assisting the proof of the correctness and degree of optimality of our proposal.

Proposition 1: The WCQD Q_1 for the first priority level (highest priority) is given by $Q_1 = \sum_{\forall \, f \in \mathcal{F}} b_f / C$ when there is a single priority level or $Q_1 = (\sum_{\forall \, f \in \mathcal{F}_1} b_f - max_{\forall f \in \mathcal{F} \setminus \mathcal{F}_1} l_f)/C$ when there are two or more priority levels. Moreover, Q_1 is the lowest WCQD in the ATS instance, i.e., $Q_1 < Q_p \, \forall \, p \in [2, P]$.

Proof: Q_1 can be directly derived from (1). From (1), the aggregated burstiness of any priority level $p \in [2, Q]$ will include the aggregated burstiness of level 1 and by definition $l_f \leq b_f \forall f \in \mathcal{F}$. Also, the effective capacity of $p \in [2, Q]$ is reduced by the aggregated committed rate in level 1. Then, it always holds that $Q_1 < Q_p \, \forall \, p \in [2, P]$. ■

Proposition 2: Decreasing one priority level from p to $p+1$ of any flow f will increase its WCQD, but reduce or does not affect the WCQD of the rest of flows. Equivalently, increasing the priority level of any flow f will decrease its WCQD, but increase or does not affect the WCQD of the rest of flows.

Proof: From (1), lowering the priority level of a flow f will reduce the aggregated burstiness of the priority level p it was originally accommodated by b_f. Since by definition

$b_f \geq l_f^{(max)}$, the WCQD of p is reduced. For the new priority level $p+1$ of f, the aggregated burst size will remain the same, but its effective capacity $C - \sum_{k=1}^{p} r^{(k)}$ will increase by r_f, thus, decreasing the WCQD of $p+1$. For priority levels $k > p+1$ or $k < p$, the WCQD does not change. On the other hand, the maximum aggregated burst size seen by f remains the same, but its effective capacity is reduced by $\sum_{f \in \mathcal{F}_p} r_f$, thus increasing its WCQD. Last, increasing the priority level of a flow f is equivalent, in terms of the resulting WCQDs experienced by the flows, to keep the same priority level for f and decrease one priority level for the rest of the flows. ■

Proposition 3: If, in the highest priority level ($p = 1$), the most lenient WCQD requisite $R_{f_{F_1}}$, i.e., $R_{f_{F_1}} \geq R_f \, \forall \, f \in \mathcal{F}_1$, which is imposed by the flow $f_{F_1} = f_{|\mathcal{F}_1|}$, is not fulfilled, i.e., $Q_1 > R_{f_{F_1}}$, then, problem (2) has no satisfiable solution.

Proof: From *Proposition 1*, $Q_1 < Q_p \, \forall \, p \in [2, P]$. From *Proposition 2*, decreasing the priority level of f_{F1} will increase its WCQD. On the other hand, decreasing the priority level of any flow $f \in \mathcal{F}_1 \, s.t. \, f \neq f_{F1}$ to reduce the Q_1 is neither possible because the WCQD of f will increase (*Proposition 2*) and from the premises of the proposition $R_{f_{F_1}} \geq R_f$, thus, R_f would not be met. ■

Proposition 4: If currently $Q_p \leq R_f \, \forall \, f \in \mathcal{F}_p$ and $\forall \, p \in [2, P-M]$, $\mathcal{F}_p == \emptyset \, \forall \, p \in [P-M+1, P]$, and we decrease M priority levels $\forall \, f \in \mathcal{F} \setminus \mathcal{F}_1$, then, we can freely distribute the flows originally allocated to $p = 1$ among the levels $p \in [1, M+1]$ and the requisites of the rest of the flows will be still met, i.e., $Q_p \leq R_f \, \forall \, f \in \mathcal{F}_p$ and $\forall \, p \in [2+M, P]$.

Proof: From (1), decreasing M priority levels for all the flows originally allocated in $p \in [2, P-M]$ will keep the same WCQD for them as $\mathcal{F}_p == \emptyset \, \forall \, p \in [P-M+1, P]$. Then, no matter the prioritization we consider for the flows originally allocated in $p = 1$ among the levels $p \in [1, M+1]$, also from (1), the WCQD will remain the same for all the flows originally allocated in $p \in [2, P-M]$. ■

B. Algorithm

The proposed ATS flow prioritization algorithm is shown in Algorithm 1. The goal of the algorithm is to find a satisfiable prioritization, if at least one exists, for the set of flows $\mathcal{F}$ at a given ATS instance according to the optimization program (2). To that end, it iterates ($lines \; 7-20$) until either a feasible solution is found, i.e., the delay requisites for all the flows are met while the link utilization is lower than 100% ($line \; 10$), or the problem infeasibility is determined ($line \; 17$). Please refer to *Propositions 1* and *3* for the rationale behind the latter algorithm exit condition.

At each iteration, first, the algorithm checks whether the WCQD requisites for all the flows allocated to the second priority level $\mathcal{F}_2$ are met ($line \; 8$). Please note that this condition is always met at the very first iteration as $\mathcal{F}_2$ initially equals the empty set. If the condition is met, which is verified using (1), the algorithm checks, again using (1), whether the WCQD requisites for all the flows allocated to $\mathcal{F}_1$ are met. If so, a feasible solution is found ($line \; 10$) and the algorithm finishes. Otherwise, the algorithm creates a new set $\mathcal{F}_k$ if needed and decreases the priority level for all the flows by one, leaving the set $\mathcal{F}_1$ empty ($lines \; 12-13$). We refer hereinafter to this process as partition k or kth. The reason to follow the operation described above is that, once the algorithm finds a satisfiable prioritization for the flows allocated to the current priority levels 2 to k, the highest priority level can be further partitioned to find a feasible solution without affecting the WCQDs of the current priority levels 2 to k (*Proposition 4*).

If conditions in $line \; 8$ or $line \; 9$ are not met, then, the algorithm moves the flow f^* with the most stringent WCQD requisite currently in priority 2 to priority 1, i.e., it increases the priority of f^* ($lines \; 15-16$). Nonetheless, if it turns out that f^* is the last flow in $\mathcal{F}_2$, the algorithm realizes that the problem has no solution ($line \; 17$).

C. Algorithm Analysis

This section includes the analysis of the proposed prioritization algorithm for ATSs detailed in Section IV-B. More precisely, we rely on the principle of mathematical induction to formally prove the theorem stated next.

Theorem 1: Algorithm 1 finds always a satisfiable solution for the ATS prioritization problem (2) if any exists and the solution found is optimal for that problem, i.e., it minimizes the number of priority levels used in the ATS instance.

Proof: Let k denote the current partition index as in Algorithm 1 to find a solution for prioritizing a set of TSN flows $\mathcal{F}$. As previously defined, a partition is the process carried out by Algorithm 1 for partitioning the current highest priority level into two. That is, decreasing the priority level of all the flows by one and, after, moving as many flows from $p = 2$ to $p = 1$ according to Algorithm 1 (see $lines \; 7-20$). Last, observe that for $k = 1$ Algorithm 1 just checks whether a satisfiable prioritization exists for a single priority level, i.e.,

Algorithm 1 ATS Prioritization Algorithm

```
1:  Problem_Solved = 1;                                   ▷ BC_O1
2:  No_Solution = 2;                                      ▷ BC_O2
3:  Searching_Solution = 3;                               ▷ BC_O3
4:  Initialize F_1 ← F; F_2 ← ∅; k = 1;
5:  function PrioritizeFlows(F_1, F_2, k)
6:      prob_status = Searching_Solution;
7:      while prob_status == Searching_Solution do
8:          if Q_2 ≤ R_f ∀f ∈ F_2 then
9:              if Q_1 ≤ R_f ∀f ∈ F_1 then
10:                 return Problem_Solved;
11:             end if
12:             k++; F_k = {};
13:             F_p ← F_{p-1} ∀p = [2, k]; F_1 ← ∅;
14:         end if
15:         f* ← arg min_{f∈F_2} R_f;
16:         F_2 ← F_2 \ f*; F_1 ← F_1 ∪ {f*};
17:         if F_2 == ∅ then
18:             return No_Solution;
19:         end if
20:     end while
21: end function
```

$Q_{spl} \leq R_1$. Trivially, if Algorithm 1 finds a solution for $k = 1$, the main hypothesis holds true.

BASE CASE (k=2): For the base case Algorithm 1 may result in three possible outcomes: i) a satisfiable solution is found for two priority levels (BC_O1) ii) the prioritization problem has no solution (BC_O2), and iii) further partitions are required to find a potential feasible prioritization (BC_O3). For BC_O1, the hypothesis holds true as Algorithm 1 has previously explored the single priority level configuration and determined it is unfeasible. For $k = 2$, output BC_O2 is issued when the flow f_F verifying that $R_{f_F} \geq R_f \, \forall f \in \mathcal{F}$ cannot be even accommodated in a single priority level. That is because, in $k = 1$, the algorithm could not accommodate all the flows in a single priority level and in $k = 2$ the algorithm has move all the flows to $p = 1$ except f_F without fulfilling R_{f_F} (see in *lines 8* and *17*). Then, from *Proposition 3*, the problem (2) has no solution and the hypothesis still holds true.

It remains to show that for BC_O3 no satisfiable solution for two priority levels exists. For this output, the WCQD requisite of f_1 (the most stringent constraint), allocated to $p = 1$, is not fulfilled as $R_{f_1} \leq R_f \, \forall f \in \mathcal{F}$ and, in $k = 2$, BC_O3 happens when condition in *line 9* is unfulfilled. From *Proposition 2*, increasing the priority of any of the flows allocated to $p = 2$ only contributes to the nonfulfillment of R_1. On the other hand, if moving any of the flows $f \neq f_1$ in $p = 1$ to $p = 2$ would result in a feasible solution, then, from *Proposition 2*, decreasing the priority of any other flow s in $p = 1$ verifying that $R_s \geq R_f$ will also result in a feasible solution. Last, observe that Algorithm 1 accommodates as many flows with the most lenient requisites as possible in $p = 2$ at each partition, thus minimizing Q_1. Considering all above, we can conclude that no satisfiable solution exists for two priority levels if the Algorithm 1 status is BC_O3 at the end of $k = 2$. Then, the hypothesis still holds true.

INDUCTION CASE (k=n+1): For partition n, Algorithm 1 has distributed the flows with the most lenient requisites in $p = [2, n]$ and their requisites are met. Then, if the requisite of f_1, accommodated in $p = 1$, is not met further partitions are needed. The induction method allows us to assume that the hypothesis holds true for iteration n, i.e., the prioritization of the flows in $p \in [2, n]$ is satisfiable and it is the one requiring the minimum number of priority levels (induction hypothesis). For $k + 1$, similar to the "BASE CASE", Algorithm 1 will keep partitioning $p = 1$ in search of a feasible solution. From *Proposition 4*, the analysis carried out for the "BASE CASE" is also valid for the "INDUCTION CASE". Considering this fact together with the induction hypothesis, we can conclude that the main hypothesis holds true. Observe, after $k = n + 1$, Algorithm 1 might keep making partitions until letting f_1 alone in $p = 1$. At that point, the feasibility of the problem is easily checked from *Proposition 3*. ■

Last, note that Algorithm 1 might output a feasible prioritization that requires a number of priority levels higher than the one supported by the respective ATS instance. In this case, the solution would be unfeasible due to the aforementioned constraint. However, that does not affect the analysis.

V. Results

In this section we provide the experimental results to show the scalability, optimality, and correctness of our proposal.

A. Experimental Setup

First, we devised a compound traffic model, summarized in Table II, based on [11], [13], [14] and references therein to realistically capture in our experiments the typical per-flow traffic demands and delay requisites in industrial scenarios. Each performed experiment consisted in finding the flow prioritization in an ATS instance for $F = |\mathcal{F}|$ flows. The prioritization problem was solved using the brute force algorithm and Algorithm 1 for comparison. Brute force consists of checking all the possible prioritizations for the flows in $\mathcal{F}$, and selecting that one requiring the minimum number of priority levels. Both algorithms were developed in Matlab and run in a server with Intel(R) Core(TM) i7-6700K Central Processing Unit (CPU) at 4.00GHz with 4 cores and 32 GB of RAM. Since problem (2) fast becomes intractable with the scenario scale when it is solved using brute force, we decided to do a per IEEE 802.1Q Traffic Class (TC) prioritization. Specifically, each service in Table II was mapped onto a TC and the respective Priority Code Point (PCP) (see sixth column in Table II). For each experiment, the number of flows $F^{(q)}$ considered for each TC q was proportional to its per-flow expected committed data rate (second column in Table II), i.e., $F^{(q)} = E[r_q]/(\sum_{k=1}^{7} E[r_k]) \cdot F$. Last, the traffic characteristics and delay requisite of each individual flow were uniformly sampled from the ranges provided in Table II according to the TC it belongs to. Please observe that the procedure described above results in seven TCs to be prioritized, each characterized by: i) a committed data rate and a burst size that correspond to the sum of the committed data rate and burstiness of all of its flows, respectively; ii) a maximum frame size which is determined as the maximum frame size among all of its flows; and iii) a delay requisite established as the most stringent delay requirement among all the flows. Assuming the traffic conforms to the committed data rate and burstiness the ATS produces zero packet losses. For

TABLE II
PER-SERVICE FLOW CHARACTERISTICS.

Services	r_q (Mbps)	b_q (packet)	D_q (ms)	l_Q (KBytes)	**PCP**
Cyclic-Synchronous	$8-0.8$	$1-4$	$1-0.5$	$1-0.05$	6
Mobile Robots	< 10	$1-4$	$500-1$	$0.25-0.04$	3
Cyclic-Asynchronous	$0.2-4e^{-3}$	$1-4$	$20-2$	$1-0.05$	5
Events: Control	> 12	$1-4$	$50-10$	$0.2-0.1$	4
Augmented Reality	$20-10$	$1-4$	10	$1.5-0.03$	2
Network Control	$8e^{-3}-4e^{-3}$	$1-4$	$1000-50$	$0.5-0.05$	7
Config. & Diagnostics	2	$1-4$	$100-10$	$1.5-0.5$	1

each value of F considered, we executed 100 independent realizations with different flow characteristics in each realizations. The execution times measurements reported are the average of all those runs resulting in the same number of TCs.

B. Performance Evaluation

Fig. 2 depicts the comparison of the average execution time exhibited by the brute force (labeled as 'Brute Force') and our heuristic-based (labeled as 'Developed algorithm') algorithms as a function of the number of TCs to be prioritized.

As observed, for the considered range of TCs, the results show that the execution time of the brute force algorithm exhibits an exponential growth whereas the our proposal scales well. For instance, for prioritizing seven TCs the brute force's execution time is six orders of magnitude higher than our proposal. This makes unfeasible to use the brute force algorithm to carry out a per-flow prioritization in the ATS.

Fig. 3 depicts the prioritization outputted by our solution for different scenarios. Each scenario includes a given number of flows (x-axis) grouped into TCs as explained in the previous subsection. For each value of F, 100 independent realizations were carried out, each sampling the flow features according to the ranges provided in Table II. The line labeled as '%' represents the number of realizations of each scenario in which a feasible solution were found. Similarly, each bar, labeled as 'Prior P', represents the percentage of realizations requiring at least P priority levels. As expected, the higher the number of flows in the scenario, which translates into higher utilization of the ATS link, the higher the probability of not finding a satisfiable solution, even if the number of priority levels is increased. Similarly, the minimum number of priority levels required increases with the traffic load. The most remarkable result of this experiment is that both our proposal and the brute force algorithm outputted exactly the same prioritization for all the experiments, thus validating our proposal's operation and optimality. These results support that our solution finds a satisfiable solution if it exists and that solution requires the lowest number of priority levels as stated in Section IV.

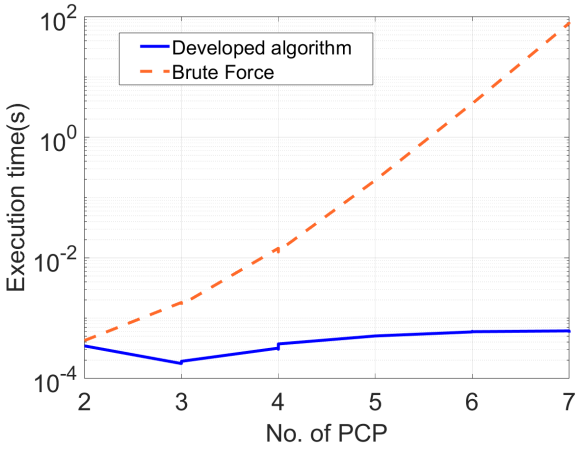

Fig. 2. Algorithm execution time.

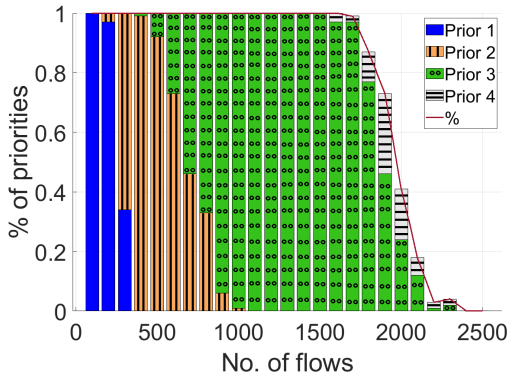

Fig. 3. Priority assignment for different flows.

VI. Conclusion

In this paper, the ability of the developed prioritization algorithm to obtain a feasible solution with the lowest WCQD in TSN has been evaluated. We have formally formulated the prioritization problem and the resolution through our proposal. Our algorithm and brute force have been compared with industrial traffic. The outcomes demonstrate that our approach scales correctly, achieving an execution time that is six orders of magnitude less than that of brute force and achieving the same prioritizing outcomes for both algorithms.

References

[1] J. Specht and S. Samii, "Synthesis of queue and priority assignment for asynchronous traffic shaping in switched ethernet," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 178–187.

[2] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Learnet: Reinforcement learning based flow scheduling for asynchronous deterministic networks," in *2020 IEEE Int. Conf. on Commun. (ICC)*, 2020, pp. 1–6.

[3] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Optimization of flow allocation in asynchronous deterministic 5g transport networks by leveraging data analytics," *IEEE Trans. Mob. Comput.*, pp. 1–1, 2021.

[4] A. Nasrallah *et al.*, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Commun. Surveys Tutorials*, vol. 21, no. 1, pp. 88–145, Firstquarter 2019.

[5] "Ieee draft standard for local and metropolitan area networks–media access control (mac) bridges and virtual bridged local area networks amendment: Asynchronous traffic shaping," *IEEE P802. 1Qcr/D2.1*, pp. 1–152, Feb. 2020.

[6] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conf. on Real-Time Systems (ECRTS)*, July 2016, pp. 75–85.

[7] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2721–2733, Dec. 2018.

[8] J. Prados-Garzon and T. Taleb, "Asynchronous time-sensitive networking for 5g backhauling," *IEEE Netw.*, vol. 35, no. 2, pp. 144–151, 2021.

[9] J. Prados-Garzon, L. Chinchilla-Romero, P. Ameigeiras, P. Muñoz, and J. M. Lopez-Soler, "Asynchronous time-sensitive networking for industrial networks," in *2021 Joint European Conf. on Netw. and Commun. & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 130–135.

[10] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001.

[11] 3GPP TS22.104 V17.4.0. (2020) Service Requirements for Cyber-Physical Control Applications in Vertical Domains.

[12] "IEEE Draft Standard for Local and metropolitan area networks–Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping," *IEEE P802.1Qcr/D2.1, Feb. 2020*, pp. 1–152, 2020.

[13] "Integration of 5g with time sensitive networking for industrial communications," White Paper, 5G ACIA, Feb. 2021.

[14] "A 5g traffic model for industrial use cases," White Paper, 5G ACIA, Nov. 2019.