Introducing Guard Frames to Ensure Schedulability of All TSN Traffic Classes

Daniel Bujosa*, Julian Proenza†, Alessandro V. Papadopoulos*, Thomas Nolte*, Mohammad Ashjaei*
*Mälardalen University, Västerås, Sweden
† University of the Balearic Islands, Palma, Spain

***Abstract*—Offline scheduling of Scheduled Traffic (ST) in Time-Sensitive Networks (TSN) without taking into account the quality of service of non-ST traffic, e.g., time-sensitive traffic such as Audio-Video Bridging (AVB) traffic, can potentially cause deadline misses for non-ST traffic. In this paper, we report our ongoing work to propose a solution that, regardless of the ST scheduling algorithm being used, can ensure meeting timing requirements for non-ST traffic. To do this, we define a frame called Guard Frame (GF) that will be scheduled together with all ST frames. We show that a proper design for the GFs will leave necessary porosity in the ST schedules to ensure that all non-ST traffic will meet their timing requirements.**

***Index Terms*—Time-Sensitive Networking (TSN), offline schedule synthesis, AVB schedulability.**

I. Introduction

Since its introduction in 2012, Time Sensitive Networking (TSN) has become one of the most relevant sets of standards for the time-sensitive transmission of data. TSN provides Ethernet with hard and soft real-time traffic transmission on the same network, deterministic zero-jitter and low-latency transmission, precise clock synchronization, and fault tolerance mechanisms. Thanks to these features, TSN has become increasingly significant in areas such as automotive [1], automation [2], and energy distribution [3].

The relevant features related to traffic transmission consist of a set of traffic shaping mechanisms. These mechanisms are Strict Priority transmission, Time-Aware Shaping (TAS), and Credit-Based Shaping (CBS), to name the major ones. According to the Strict Priority transmission, each frame belongs to one of the eight different priority levels. In addition, depending on the traffic shaper that is applied, one priority can belong to one of the three basic classes of TSN traffic, including Scheduled Traffic (ST), Audio-Video Bridging (AVB) traffic, and Best Effort (BE) traffic. The ST commonly has the highest priorities and is used for the transmission of time-triggered traffic with strict timing requirements which are scheduled offline. The offline schedule specifies in what time slot, also called *transmission window*, the frame should be transmitted in each link to meet its timing requirements. This makes ST fully deterministic with zero jitter in the transmission. To do this, the ST uses TAS which is able to interrupt any other communication, regardless of its priority, to guarantee the ST transmission without any interference. The AVB traffic commonly has middle priorities and makes use of the CBS which limits the amount of bandwidth it can use to improve the Quality of Service (QoS) of lower-priority traffic. Finally, the BE traffic commonly has the lowest priorities and does not offer any real-time guarantees.

Although ST transmission via TAS brings determinism, finding a proper schedule on a multi-hop TSN network is a daunting task, which is known to be an NP-complete problem [4]. In particular, TAS brings a high flexibility compared to similar time-division communication technologies, e.g., FlexRay, since in TAS a frame can be freely scheduled over the timeline instead of allocating a dedicated window for all time-triggered traffic. The high flexibility of ST frame scheduling comes with a cost of schedule synthesis complexity and several solutions have been proposed for that, e.g., constrained programming-based solutions [4] and heuristics [5].

However, most of the proposed solutions to schedule ST are customized to obtain optimum schedules for ST with very little attention, if any, on the QoS of non-ST traffic (i.e., periodic and sporadic time-sensitive traffic such as AVB traffic, functional safety data, or alarms). Certain schedules for ST can potentially cause missing deadlines for non-ST traffic. Few works have addressed such a problem, e.g., proposing slacks after ST frames to ensure spaces for BE traffic [6] or considering delay analysis of AVB inside the ST schedule [7]. Nevertheless, to the best of our knowledge, the solutions are customized for a specific ST scheduling algorithm and none of them guarantee meeting deadlines for all non-ST traffic.

In this paper, we report our ongoing work to propose a solution, regardless of the scheduling algorithm being used, to ensure that while we provide a schedule for ST, the non-ST traffic meet their timing requirements. In order to do that, we propose to use a fictitious frame, called Guard Frame (GF), to be scheduled together with the other ST frames and removed prior to deployment on the TSN network. We design the GF in such a way that it restricts the ST distribution throughout the schedule and generates a new ST distribution with a porosity such that non-ST traffic will meet their timing requirements. This restriction guarantees that, without the need of reserving all the necessary resources for each non-ST frame, if a feasible schedule for the restricted ST is obtained, the non-ST frames will never experience deadline misses regardless of their arrival time. Note that the solution is different than simply reserving bandwidth for non-ST traffic since, due to the TAS operation, all bandwidth not used for the ST is already reserved for the non-ST traffic.

II. Related Work

ST schedulers can be categorized into two groups: (i) those based on Satisfiability Modulo Theorems (SMT) and Optimization Modulo Theorems (OMT), and (ii) heuristics and metaheuristics. SMT solvers explore variable combinations to find optimal schedules by verifying logical statements' satisfiability. OMT solvers, an extension of SMT, optimize variables based on user-defined objectives to improve scalability [8]. On the other hand, heuristic schedulers synthesize schedules by assigning time slots frame by frame according to a pre-established order. While faster, they offer lower schedulability without guaranteeing optimality. Most of the work on ST schedulers is based on the work presented in [9]. Among them, [10] developed a GCL synthesis scheduler for ST in TSN, and [11] introduced TSNSched, an SMT-based ST scheduler utilizing TSN network models. Heuristic approaches, like [12], [13], [14], focused on heuristics and GA-based solutions to overcome high processing times of constraint programming-based schedulers, although no comparisons were made with SMT-based solutions. Additionally, [15] proposed a Tabu search-based approach, prioritizing ST frame scheduling for enhanced bandwidth utilization.

The above-presented schedulers only take into account the ST to synthesize the schedule. However, this may limit, among others, the QoS of non-ST traffic or the reconfigurability of the network. For this reason, different authors have proposed porous ST schedulers which leave blank spaces between ST transmission windows. For example, in [16] the authors presented strategies for the synthesis of porous schedules by adding blank slots during the scheduling or after it. Moreover, the solution in [17] focuses on generating porous link schedules to accommodate possible link failures by applying a link repair post-processing procedure to find alternative routes. Finally, the solution in [6] proposes to produce porous schedules by scheduling the ST frames as close as possible to the deadlines, maximizing the distance between ST transmission windows, or distributing the ST transmission windows homogeneously over the hyper-period.

However, the solutions above show some limitations. First, all of them are based on SMT or OMT solvers, so they are time-consuming and not scalable. Moreover, since the goal is to sparse, they can hardly optimize latency or other timing metrics. In addition, most of them do not take into account the specific requirements of non-ST traffic, thus the sparsity generated may be higher than necessary. In this paper, we propose an initial solution that (i) can be applied on any ST scheduler; (ii) can guarantee that non-ST frames (e.g., AVB frames) will meet their timing requirements.

III. Proposed Solution

The main problem with scheduling ST without taking into account non-ST traffic is that, although many schedules meet the requirements of ST, not all of them may be suitable to meet the requirements of non-ST traffic. This is because non-ST frames have a maximum amount of interference from ST that they can experience without missing their deadlines. Therefore, we propose the design of a GF per link which is scheduled along with the other ST frames and will be removed from the schedule before its deployment. This ensures that the necessary time slots will be available for the non-ST frames to meet their deadlines regardless of their arrival time.

For example, imagine a network consisting of a single link through which 5 ST frames each with size of 1 time unit (tu) and period of 10 tu, and 1 non-ST frame of size 1 tu and period 5 tu are sent through. We assume that their deadlines are equal to their periods. In this case, if the ST is scheduled without considering the requirements of the non-ST traffic, and in addition we seek to minimize latency, the 5 ST frames will be scheduled to occupy the first 5 time slots of the hyper-period. This schedule is shown in Fig. 1 which leads to the non-ST frame meeting its deadline if its arrival time is not at the beginning of the hyper-period. However, if the non-ST frame arrives at the beginning of the hyper-period it will miss its deadline (as shown in Fig. 1).

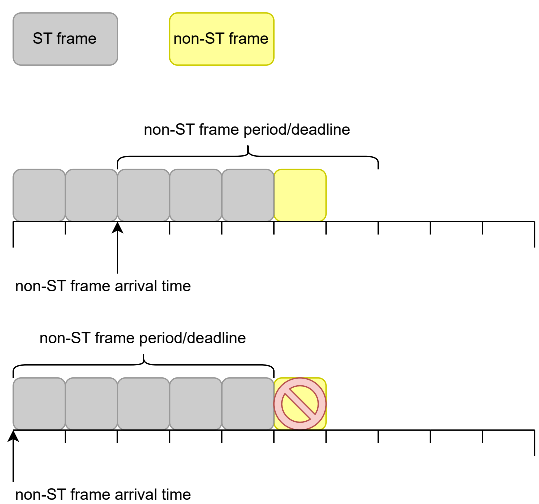

Fig. 1. Example of ST maximum interference.

In order to prevent the deadline misses for the non-ST frame in this example, as shown in Fig. 2, it would be enough to create a GF of size 1 tu and period 5 tu to free the necessary slot for the non-ST frame to meet its deadlines. Following we describe how the GFs should be designed to prevent the deadline misses for the non-ST frames.

Consider a network consisting of $\mathcal{L}$ unidirectional links $l \in \mathcal{L}$ through which $\mathcal{F}$ non-ST frames $f \in \mathcal{F}$ will be transmitted through routes $f.\zeta = \{l_i..l_n\} \in \mathcal{L}$ where l_i is the link connected to the source and l_n is the link connected to the destination. To design the GF of each link, i.e., determine its size and period, all we need to know is the deadline of each non-ST frame ($f.d$) since it corresponds to the maximum interference that each non-ST frame f can experience along the path from the source to the destination to meet its deadline. This maximum interference is divided into two sources of interference. The first source of interference for frame f is called Maximum ST Interference ($MSTI_f$). The $MSTI_f$ is the maximum ST interference that a frame f experiences

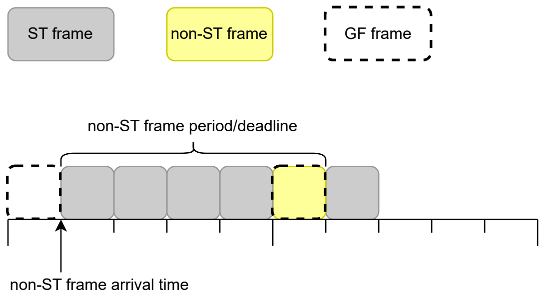

Fig. 2. Example of GF preventing ST maximum interference.

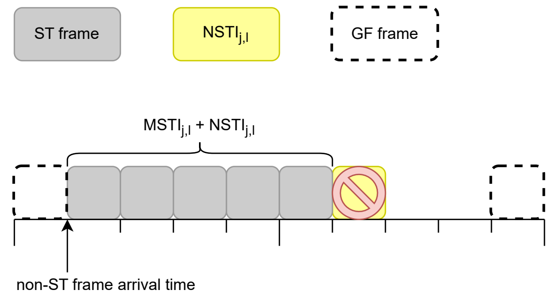

Fig. 3. ST interference with variable offsets.

which depends on the ST schedule. This value is per frame for the entire frame route $f.\zeta$, hence it will be distributed among the different links that conform to it $l \in f.\zeta$. The second source of interference for frame f on link $l \in f.\zeta$ is called Non-ST Interference ($NSTI_{f,l}$). The $NSTI_{f,l}$ includes, apart from transmission time of frame f, the maximum interference that it can suffer from other causes such as interference from unscheduled higher-priority traffic, interference from the same priority, blocking of lower-priority traffic, and other delays such switch processing. The value of $NSTI_{f,l}$ can be either dependent or independent of the ST interference. If the $NSTI_{f,l}$ value is independent of the ST interference, it can be calculated in advance; otherwise, it will be necessary to know the $MSTI_f$ of each link to calculate it. Note that,

$$f.d = MSTI_f + \sum_{l \in f.\zeta} NSTI_{f,l} \tag{1}$$

Above, we define that the deadline of frame f is equal to all maximum interference by ST and non-ST traffic. Note that $MSTI_f$ and $\sum_{l \in f.\zeta} NSTI_{f,l}$ can be derived by adapting any worst-case delay analysis, e.g., the response time analysis given for TSN [18], which is the ongoing work.

We will now outline the requirements that must be met by the GFs to ensure that all non-ST frames do not exceed their $MSTI_f$. We will start by considering schedulers with fixed offsets and then extend the solution to variable offsets. In fixed offset scheduling, ST frames are scheduled with constant delays relative to the start of the period. For example, an ST frame with a period of 5 tu can be scheduled to be transmitted at times 0, 5, and 10 (with offset 0) or at times 2, 7, and 12 (with offset 2). This means that the offset is fixed for all ST arrival instances in the timeline. In contrast, variable offset scheduling allows for more flexible scheduling which allows setting different offsets for each ST instance. For example, transmitting the ST frame at 2, 5, and 11, giving offsets of 2, 0, and 1 for 3 instances of the ST frame. Our solution considers both fixed and variable ST offsets scheduling.

First of all, we must distribute the $MSTI_f$ of each frame among the links that conform to its route to get $MSTI_{f,l}$. This can be done uniformly, inversely proportional to the ST load of each link, or by any other allocation method. Next, we must specify the GF size and period for each link in a way that for any given time interval $MSTI_{f,l} + NSTI_{f,l}$ the amount of ST interference does not exceed $MSTI_{f,l}$. To this end, the GF period and deadline of each link must be smaller than or equal to the smallest sum of $MSTI_{f,l}$ and $NSTI_{f,l}$, while its utilization must be greater than or equal to the largest

$$\frac{NSTI_{f,l}}{(MSTI_{f,l} + NSTI_{f,l})} \tag{2}$$

Note that the GF is a fictitious frame and is not queued, hence it should not be affected by any queuing condition during its scheduling, only by the link scheduling allocation.

Previous GF definition ensures that no frame f traversing the link l will experience an ST interference greater than $MSTI_{f,l}$ in any time interval $MSTI_{f,l} + NSTI_{f,l}$ regardless of its arrival time if the offsets are fixed. However, if the offsets are variable, depending on the arrival time, a non-ST frame may experience more ST interference than $MSTI_{f,l}$. Retrieving the example of Figs. 1 and 2 we can now identify how $MSTI_{f,l}$ and $NSTI_{f,l}$ are 4 tu and 1 tu, respectively. In Fig. 3 we can see how the same GF with period 5 tu and size 1 tu does not guarantee that the non-ST frame will meet its deadline. The variable offsets result in time instants in which the ST interference exceeds the $MSTI_{f,l}$. This is because for certain arrival instants, the first and last instances of the GF can be scheduled in such a way that both fall outside the time interval $MSTI_{f,l} + NSTI_{f,l}$ causing more ST interference than allowed. In this case, to ensure that the $MSTI_{f,l}$ is never exceeded, it is necessary to increase the utilization of the GF.

Given a GF of period T and size C that meets the conditions previously stated for a fixed offsets scheduler, we define the coefficient between the time interval $MSTI_{f,l} + NSTI_{f,l}$ and T as N:

$$N = \frac{MSTI_{f,l} + NSTI_{f,l}}{T} \tag{3}$$

For a new GF of period T and size C' to always limit the interference from ST to $MSTI_{f,l}$ with variable offsets, the following condition must be met for all f in the link:

$$\frac{(N-1) \times C' + \max(2 \times C' - T, 0)}{MSTI_{f,l} + NSTI_{f,l}} \geq \frac{C}{T} \tag{4}$$

The above inequality has two parts. The right side represents the utilization that GF should have at all times to meet its objective, while the left side represents the utilization it would have in the worst-case scenario in a given time interval

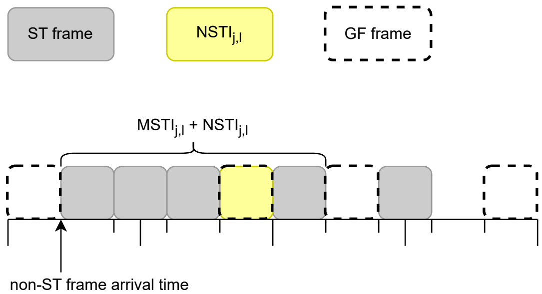

Fig. 4. Example of GF preventing ST maximum interference with variable offsets.

$MSTI_{f,l} + NSTI_{f,l}$ with variable offsets. Operating with Equation (3) we obtain:

$$(N-1) \times C' + max(2 \times C' - T, 0) \geq N \times C \quad (5)$$

that in case $2 \times C' - T$ is negative the result is:

$$C' \geq \frac{N}{N-1} \times C \quad (6)$$

This new value of C' ensures that the GF meets the necessary schedule porosity to meet non-ST traffic time requirements for an ST scheduler with variable offsets. Note that variable offsets are less restrictive for the ST scheduler but it implies a more bandwidth-intensive GF per link which is another factor that can limit the schedulability. Returning once again to the previous example, we see that by applying the solution we obtain a GF of period 5 tu and size 3 tu. However, the utilization of this new GF together with the ST frames exceeds 100% so to alleviate the load we will reduce the period to 2.5 tu. Finally, applying again the solution we obtain a GF of period 2.5 tu and size 1 tu which, as shown in Fig. 4, guarantees that the non-ST frame never misses its deadline.

IV. Conclusions and Future Work

In this work, we have proposed a frame called Guard Frame (GF). The solution consists of the design of a fictitious frame per link that, after being scheduled together with the rest of the ST frames and eliminated, generates a schedule with a porous distribution of the ST frames that guarantees meeting non-ST traffic deadlines. This solution is applicable to any type of ST scheduling algorithm yet it requires the calculation of the maximum interference that each frame can experience without missing its deadline. In our ongoing work, we will develop a method to calculate the maximum ST interference that any non-ST frame can experience without missing its deadline by modifying one of the existing worst-case delay analysis for TSN networks. In addition, we will implement the GF in a series of experiments with an existing scheduler to evaluate its performance in terms of schedulability.

Acknowledgment

This research is supported by the Swedish Knowledge Foundation (KKS) under the FIESTA project and the Swedish Governmental Agency for Innovation Systems (VINNOVA) under the DESTINE and PROVIDENT projects. Julian Proenza is supported by Grant pid2021-124348ob-i00, funded by MCIN/AEI/ 10.13039/501100011033 / ERDF, EU.

References

[1] S. Samii and H. Zinner, "Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 62–68, 2018.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald, "Utility Applications of Time Sensitive Networking White Paper (D3)," IEEE, Tech. Rep., 2018.

[4] M. L. Raagaard and P. Pop, "Optimization algorithms for the scheduling of IEEE 802.1 Time-Sensitive Networking (TSN)," Tech. Univ. Denmark, Tech. Rep., 2017.

[5] D. Bujosa, M. Ashjaei, A. V. Papadopoulos, T. Nolte, and J. Proenza, "HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities," in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, 2022, pp. 70–80.

[6] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "Synthesising schedules to improve QoS of best-effort traffic in TSN networks," in *29th International Conference on Real-Time Networks and Systems*, 2021, pp. 68–77.

[7] A. Berisa, L. Zhao, S. Craciunas, M. Ashjaei, S. Mubeen, M. Daneshtalab, and M. Sjödin, "AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption," in *The 30th Inter. Conference on Real-Time Networks and Systems*, June 2022.

[8] N. Bjørner, A.-D. Phan, and L. Fleckenstein, "νz-an optimizing SMT solver," in *Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2015, pp. 194–199.

[9] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *IEEE Real-Time Systems Symposium (RTSS)*, 2010, pp. 375–384.

[10] R. S. Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1 Qbv gate control list synthesis using array theory encoding," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 13–24.

[11] B. Schneider, "Automatic network configuration for real-time, distributed industrial automation systems," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 629–634.

[12] M. Pahlevan and R. Obermaisser, "Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks," in *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*, vol. 1, 2018, pp. 337–344.

[13] M. Pahlevan, J. Schmeck, and R. Obermaisser, "Evaluation of tsn dynamic configuration model for safety-critical applications," in *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2019, pp. 566–571.

[14] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," *ACM Sigbed Review*, vol. 16, no. 1, pp. 15–20, 2019.

[15] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[16] W. Steiner, "Synthesis of static communication schedules for mixed-criticality systems," in *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, 2011, pp. 11–18.

[17] F. Pozo, G. Rodriguez-Navas, and H. Hansson, "Schedule reparability: Enhancing time-triggered network recovery upon link failures," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 147–156.

[18] L. L. Bello, M. Ashjaei, G. Patti, and M. Behnam, "Schedulability Analysis of Time-Sensitive Networks with Scheduled Traffic and Preemption Support," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 153–171, June 2020.