# AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption

Aldin Berisa
Mälardalen University
Västerås, Sweden
aldin.berisa@mdu.se

Luxi Zhao*
Beihang University
Beijing, China
zhaoluxi@buaa.edu.cn

Silviu S. Craciunas
TTTech Computertechnik AG
Vienna, Austria
silviu.craciunas@tttech.com

Mohammad Ashjaei
Mälardalen University
Västerås, Sweden
mohammad.ashjaei@mdu.se

Saad Mubeen
Mälardalen University
Västerås, Sweden
saad.mubeen@mdu.se

Masoud Daneshtalab
Mälardalen University
Västerås, Sweden
masoud.daneshtalab@mdu.se

Mikael Sjödin
Mälardalen University
Västerås, Sweden
mikael.sjodin@mdu.se

## ABSTRACT

The Time-Sensitive Network (TSN) amendments and protocols add capabilities on top of standard 802.1 Ethernet for guaranteeing the timeliness of both (isochronous) scheduled traffic (ST) and shaped (audio-video) communication (AVB) in distributed applications. ST streams are guaranteed via an offline computed schedule controlling the time-aware gate mechanism of IEEE 802.1Qbv, while AVB real-time streams are shaped via a credit-based shaper (CBS) and scheduler with lower-priority than ST. Although the two traffic classes use different TSN mechanisms, they are interrelated as the ST traffic class schedule influences the latency of AVB traffic.

In this paper, we propose a method for the integration of the ST schedule synthesis with an analysis for the AVB class featuring IEEE 802.1Qbu frame preemption under different configurations to reduce the interference between the two classes. We first present a new worst-case response-time (WCRT) analysis for the AVB traffic class in TSN networks with preemption, considering an arbitrary number of AVB queues and different configurations for the CBS credit behavior. Then, we integrate the creation of ST schedule tables with the schedulability analysis of AVB traffic using a heuristic algorithm featuring frame preemption and a novel routing mechanism aimed at maximizing AVB schedulability. Finally, we evaluate our approach using both real-world and synthetic use cases showing the efficiency both in terms of schedule creation runtime and in terms of increasing the schedulability of lower-priority AVB traffic.

## CCS CONCEPTS

• **Computer systems organization → Dependable and fault-tolerant systems and networks**.

## KEYWORDS

Time sensitive networking, Scheduling, AVB, Network calculus.

*Corresponding author

*RTNS '22, June 7–8, 2022, Paris, France*


**ACM Reference Format:**
Aldin Berisa, Luxi Zhao, Silviu S. Craciunas, Mohammad Ashjaei, Saad Mubeen, Masoud Daneshtalab, and Mikael Sjödin. 2022. AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems (RTNS '22), June 7–8, 2022, Paris, France*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3534879.3534926

## 1 INTRODUCTION

Time-sensitive Networks (TSN) [13] introduce standardized real-time mechanisms for safety-critical communication over 802.1Q Ethernet networks. TSN is an umbrella term for a whole range of new amendments and protocols on top of IEEE 802.1, such as time synchronization (IEEE 802.1AS-rev), frame preemption (IEEE 802.1Qbu), frame replication/stream redundancy (IEEE 802.1CB), time-aware frame scheduling (IEEE 802.1Qbv), and many others.

Depending on the application type and domain, different safety-critical traffic classes with different levels of real-time requirements can coexist in the same network. The isochronous traffic class (c.f. [2, 3]) usually requires guaranteed latency and bounded (or even zero) jitter and generally coincides with the scheduled traffic (ST) class. The ST traffic is dispatched and forwarded within the network according to a static schedule table called the Gate-Control List (GCL), which is computed offline through exact algorithms (e.g., SMT- or ILP-based synthesis [6, 28, 33, 42]) or heuristics (e.g. [10, 23, 32]). Sporadic communication that requires bounded end-to-end latency but is not as critical as the isochronous traffic class (e.g., Audio-Video streams) falls into the audio-video-bridging (AVB) stream class. There can be multiple AVB traffic classes encoded in different priorities and shaped using a Constant-Bandwidth Server (CBS) mechanism. The standard approach is to use a formal analysis (e.g., network calculus [40]) for these traffic classes in order to determine (pessimistic) worst-case upper bounds on the response time of individual AVB streams.

Although the two traffic classes are handled using different mechanisms, they are interrelated as the schedule for the ST traffic class has an influence on the latency that AVB traffic experiences since, normally, the ST traffic is either placed in higher-priority queues or the timed-gates are closed for the AVB class if the ST class is already transmitting (c.f. [6]). In order to reduce this interference, the frame preemption mechanism introduced in IEEE 802.1Qbu can be employed, which allows certain traffic classes defined as

express (non-preemptable) to preempt frames of other traffic classes that are configured as preemptable within certain bounds (c.f. [1]). When AVB traffic is marked as preemptable, the latency bounds may be improved since AVB frames can better utilize the gaps between higher-priority ST frames at the expense of added preemption overhead. Most scheduling algorithms for the ST class do not consider the impact on the AVB (or lower-priority) traffic at all (e.g. [6, 22, 28]), while others attempt to create ST schedules that either guarantee the latency requirements of AVB traffic (e.g. [10]) or improve QoS metrics for lower-priority best-effort or AVB traffic (e.g. [11]). Additionally, most other methods do not consider the impact (and benefits) of adding preemption to the AVB traffic class. For many systems where the AVB traffic leads to high link utilization, it is very unlikely to find a schedule such that also AVB deadlines are fulfilled. The reduced AVB schedulability is partially due to the inherent pessimism of the WCRT analysis methods but also due to the non-preemptive mode for AVB since the only placement of ST schedule slots that is possible may not leave big enough gaps to fit unfragmented AVB frames. Hence, the schedulability of such systems may greatly benefit from allowing preemption in for AVB.

In this paper, we study the integration of the ST schedule synthesis with an analysis for the AVB class featuring preemption to reduce or limit the interference between the two classes. We first derive new worst-case response time (WCRT) bounds for the AVB traffic class in TSN networks with preemption, considering the standard credit behavior during the preemption overhead. Then, we use this new WCRT result to integrate the creation of GCL schedules for ST traffic with the schedulability analysis of AVB traffic. With our heuristic approach, we aim to synthesize correct ST schedules while at the same time guaranteeing the latency of the AVB traffic when preemption is enabled. The main contributions of the paper are as follows:

- we develop a novel Network Calculus-based analysis for TSN networks considering frame preemption with HOLD and RELEASE mechanism, which we use in our scheduling approach to improve the schedulability of AVB streams;
- we develop a heuristic algorithm based on [10] featuring frame preemption and an efficient routing mechanism to increase the chance of obtaining a better schedule for ST;
- we evaluate the newly proposed approach on a realistic use case as well as on synthetic networks to show co-schedulability of both ST and AVB traffic when preemption is enabled.

After a review of related work in Section 2, we introduce the TSN network and system model in Section 3. We develop the novel NC-based analysis for the preemptive model with Hold/Release mechanism in Section 4. We propose the heuristic algorithm for ST schedules in Section 5. We show the performance of our approach in Section 6 and conclude the paper in Section 7.

## 2 RELATED WORK

The scheduling problem for the ST traffic class of TSN has been studied extensively via heuristics in [20, 22–24, 32] or exact methods like ILP or SMT [6, 8, 28, 33, 41, 42]. These results do not consider the impact of (multiple) scheduled ST traffic classes on the AVB traffic class(es). Taking into account the impact of ST schedules on AVB traffic is a non-trivial problem since exact algorithms for synthesizing ST schedules based on, e.g., ILP or SMT solvers cannot be directly extended to include a network-calculus formulation within the first-order logic formulation and usually contain a feedback loop that guides the solver via optimization criteria [9]. Hence, while not being optimal, heuristic approaches are usually used since they can be extended easily to include analysis methods for AVB based on Network Calculus (NC) [18] or worst-case response time analysis (WCRT) [19], although most of them do not feature the preemptive model (e.g. [10, 11]). The work in [9] describes a feedback mechanism (via network calculus) for scheduling ST traffic in TTEthernet such that the deadlines of non-preemptable RC (AVB) traffic are fulfilled. Similarly, the work in [11] looks at scheduling TSN traffic such that the porosity of the ST timeline is increased, thereby increasing the QoS behavior of AVB traffic. However, the work in [11] does not consider preemption. The heuristic approach presented in [10] for AVB-aware routing and scheduling of TSN networks also does not feature the preemptive model for TSN.

The work in [1] presents simulations for the standardized and a novel preemption mode. Nevertheless, simulation results cannot be used for the schedulability analysis as the corner cases cannot be covered. Thiele et al. [31] presents the formal worst-case timing analysis for frame preemption supporting TSN/GCL+SP. In [40], authors propose the Network Calculus (NC)-based method to support both the non-preemption and the preemption without HOLD/RELEASE modes for the TSN/GCL+CBS architecture. Moreover, the NC-based analysis method under the preemption with HOLD/RELEASE mode for TSN/GCL+CBS is included in the technical report [7]. However, the credit behavior during overheads was assumed to be frozen, which is not following the standard, and the effect of overhead on credit bounds for Audio-Video-Bridging (AVB) traffic is not considered.

In [19], a schedulability analysis method has been proposed considering the frame preemption mechanism. However, the analysis assumes only two levels of AVB traffic and does not consider a general model with different combinations of credit behavior. The analysis in this paper is based on the Network Calculus model and considers the Hold and Release mechanisms under various modes for credit behavior.

Finally, in a recent work [43] preemption support is investigated for ST traffic (i.e., where scheduled queues can be defined as preemptible), showing that it may increase ST schedulability. An SMT formulation of the correctness constraints for ST classes with preemption is given and solved using exact methods and heuristics. Our work addresses the preemption of AVB traffic classes while ST queues are always configured as express. An interesting direction for future work may be to combine the approach in [43] with our method to enable preemption both for ST and AVB traffic classes.

## 3 TSN NETWORK AND DEVICE MODEL

Timing guarantees for the ST traffic class are possible in TSN via the clock synchronization protocol (IEEE 802.1AS), providing a common clock reference to all devices, and the timed-gate mechanism defined in IEEE 802.1Qbv, which enforces the transmission instants of ST streams (flows) encoded in so-called Gate-Control List (GCL) schedules. An ST or AVB stream is a real-time communication of a predefined payload size being transmitted from one talker (sender)

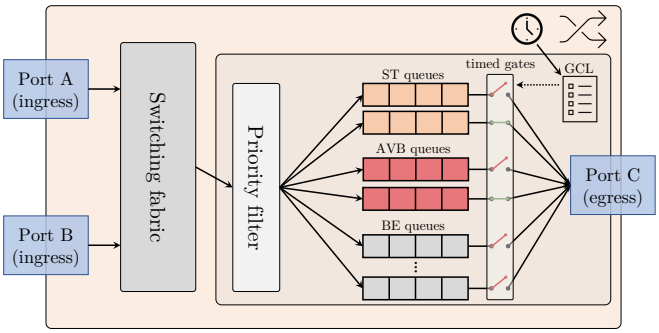


**Figure 1: Simplified TSN switch representation**

to a single or multiple listeners (receivers). While both ST and AVB streams have a requirement on the maximum latency from talker to the listener(s), ST streams have more stringent requirements in terms of jitter and determinism of transmission, usually belonging to the periodic isochronous traffic class, and AVB traffic is usually of lower criticality and has sporadic activation [2, 3].

Figure 1 presents a simplified representation of a TSN-capable switch. Streams arriving at the ingress ports (A and B) will be routed to one of the available egress ports (in our example, port C). The switching fabric selects the mapping of input to output ports for a given stream (in our simplified example representation, we only depict one egress port). After the routing to an egress port, the priority filter determines in which traffic class (queue) the frame(s) of the stream will be enqueued. Out of the 8 available queues, some will be dedicated to ST traffic (ST queues) [6] while others will enqueue AVB streams (AVB queues). The lowest-priority queues are reserved for non-critical best-effort traffic (BE queues). We note that our method works with more than two AVB classes, unlike other approaches. When using the 802.1Qbv Time-Aware Shaper (TAS), each queue has a timed-gate associated with it which precedes the transmission selection based on queue priorities. A timed-gate can be either *open (o)* or *closed (c)*, allowing or prohibiting frames from the respective queue to be sent. When multiple non-empty queues are simultaneously open, the frame(s) from the highest-priority queue will be sent. Before a gate closes, the implicit "guard band" (GB) mechanism only allows sending if there is enough time until the closing of the gate to send the next frame in the queue when considering the non-preemption mode.

The IEEE 802.1Qbu mechanism enables each traffic class (queue) to be defined as either express or preemptable. Usually, AVB and BE queues are defined as preemptable, while ST queues are configured as express. Moreover, ST schedules implement an exclusive gating between ST queues, i.e., when one ST queue is open, all others are closed, making preemption within the ST class irrelevant [6]. When using preemption, the transmission of a preemptable frame can be interrupted by an express frame within some minimum fragment size given by the minimum Ethernet packet size. When guard-band (GB) or exclusive gating is not used, the transmission latency for express frames may be reduced, and the latency of preemptable frames may increase in addition to the added preemption overhead. However, GB or exclusive gating is usually enabled to achieve the required deterministic temporal behavior of the isochronous ST class. In this case, while the ST traffic is not affected by preemption, the response time of AVB (or lower-priority preemptable) frames may be significantly improved due to fragments of the AVB frames being able to fit between ST schedule slots (c.f. Figure 2).

There are two versions for the preemption model in TSN [19]:

- The preemption with the *HOLD/RELEASE* mechanism allows an explicit "guard band" (GB) to be placed before the opening of an ST queue, thereby not allowing any lower priority traffic that is in transmission to delay ST traffic. However, the bandwidth loss is lower than under the non-preemptive model since the guarded time interval must only fit a fragment of the non-preemptable frame instead of the whole frame.
- The preemption without the *HOLD/RELEASE* mechanism allows lower-priority frames to transmit up to 123 bytes after the opening of the ST queue, thereby potentially delaying the transmission of ST frames and adding jitter to ST traffic.

The work in [1] introduced a novel preemption mode, which compared to the standard mode [14], provides a lower average response time for high-priority traffic with the downside of higher average response time for lower-priority traffic. This can be achieved by allowing higher-priority preemptable traffic to initiate transmission after a preemption by express traffic while the transmission of lower-priority traffic is already interrupted.

The assignment of streams to traffic types (ST, AVB) is defined at design time and remains fixed. For ST traffic, we also assume that the GCL is an input to the network calculus analysis, i.e., the NC analysis works with given gate opening and closing times within a known GCL period ($p^h_{\text{GCL}}$). For an AVB stream $\tau_k$, we assume that the AVB Class $\text{M}_i$, the frame size $l_k$, and the minimum interarrival time $p_k$ from the talker ES are given. Moreover, we also assume that the maximum BE frame size $l^{\max}_{BE}$ is also given.

## 4 NETWORK CALCULUS ANALYSIS FOR PREEMPTION WITH HOLD/RELEASE

In this section, we build upon the timing analysis results of the non-preemptive mode from [39] and the results from [7, 40], extending them with a timing analysis for an arbitrary number of AVB classes in the preemptive mode with both frozen and non-frozen credit during the guard band interval. As is typical in systems requiring deterministic ST traffic transmission [6, 28, 39], we assume that ST, AVB, and BE traffic are isolated in their own queues (traffic classes) and that the gates for the AVB and BE (preemptible) queues are open in a mutually exclusive fashion to the gates of the ST queues.

Frames that are smaller than or equal to $l^{\max}_{\text{nPr}} = 123$ bytes in length cannot be preempted since the minimum preemptible frame size is $l^{\min}_{\text{Pr}} = 124$ bytes. Therefore, the guard band interval in preemptive mode is reduced to $L_{\text{GB}} = (l^{\max}_{\text{nPr}} + l^{+}_{IFG})/C = (123 + 20)/C$, including fragmentation overhead and interframe gap (IFG) duration. During the guard band interval, a frame can either be preempted or finish its transmission [7]. Note that due to the gate open state of AVB traffic during the guard band, the credit of the credit-based shaper (CBS) increases. This is due to the non-frozen behavior described in the 802.1Q [14] standard (c.f. Figure 3), which has already been considered for the performance analysis with the non-preemption integration mode in [39]. A preemption overhead

**Table 1: Summary of notation.**

| Symbol | Meaning |
|---|---|
| $C$ | Physical link rate |
| $l^{\max}_{\text{nPr}}$ | The maximum non-preemptable frame size (123 bytes) |
| $l^{\min}_{\text{Pr}}$ | The minimum preemptable frame size (124 bytes) |
| $L_{\text{GB}}$ | Guard band size with HOLD/RELEASE (143 bytes/C) |
| $l^{+}_{IFG}$ | Preemption overhead (20 bytes) |
| $l^{+}_{FCS}$ | MAC DA, MAC SA, FCS frame overhead (22 bytes) |
| $l_{FCS}$ | Overhead of FCS for a frame (4 bytes) |
| $l^{\text{F,min}}_{\text{payload}}$ | Min payload (42 bytes) of the first fragment of a preemptable frame |
| $l^{\text{NF,min}}_{\text{payload}}$ | Min payload (60 bytes) for the subsequent fragments of a preemptable frame |
| $l_{OH}$ | Overhead (24 bytes) due to preemption |
| $h$ | Output port of a node |
| $\text{M}_i$ | Priority/Class of AVB traffic |
| $Q_{\text{M}_i}$ | Queue for AVB Class $M_i$ |
| $Q^{\leq i}_{AVB}$ | AVB queues with priority higher than or equal to $\text{M}_i$ |
| $n^{\text{h}}_{CBS}$ | The number of priorities for AVB traffic |
| $\beta^{\text{h,[PrH/R]}}_{\text{M}_i}(t)$ | Min-plus minimum service curve for AVB Class $\text{M}_i$ under preemption with HOLD/RELEASE |
| $\text{idSl}_{\text{M}_i}, \text{sdSl}_{\text{M}_i}$ | Idle and send slopes of the AVB class $\text{M}_i$ |
| $c^{\max}_{\text{M}_i}, c^{\min}_{\text{M}_i}$ | Upper and lower bounds of credit for AVB Class $\text{M}_i$ |
| $\alpha^{\text{h}}_{\text{ST}}(t)$ | Arrival curve for ST traffic |
| $o^{\text{h}}_i, L^{\text{h}}_{\text{ST},i}$ | Starting time and duration of $i^{th}$ ST window on port h |
| $o^{\text{h}}_{j,i}$ | Relative offset $i^{th}$ and $j^{th}$ ST windows on port h, i.e., $o^{\text{h}}_j - o^{\text{h}}_i$ |
| $p^{\text{h}}_{\text{GCL}}$ | GCL period on port h |
| $N^{\text{h}}_{\text{ST}}$ | Number of ST windows within the GCL period |
| $\alpha^{\text{h,M}_i}_{\text{OH}}(t)$ | Arrival curve with respect to preemption overheads |
| $l^{\text{h,max}}_{\text{M}_i}$ | The maximum frame of AVB Class $\text{M}_i$ on port h |
| $l^{\text{h,max}}_{\text{M}_i,\text{payload}}$ | The payload of the frame $l^{\text{h,max}}_{\text{M}_i}$ |
| $\bar{l}^{\text{h,max}}_{\text{M}_i,\text{payload}}$ | Leftover payload that has not been transmitted |
| $l^{\text{h,max}}_{\leq i}, l^{\text{h,min}}_{\leq i}$ | Max and min AVB frame size with priority $\geq \text{M}_i$ on h |
| $l^{\text{h,max}}_{< i}$ | Max AVB frame size with priority lower than $\text{M}_i$ on h |
| $l^{\text{h,M}_i}_{\text{OH},j}$ | Preemption overhead after $j^{th}$ ST window on port h |
| $n^{\text{h,M}_i}_{\text{Pr}}$ | The max preemption times of a frame of AVB Class $\text{M}_i$ |
| $L^{\text{h,M}_i}_{\text{GB},j}$ | Guard band duration before $j^{th}$ ST window on port h |
| $\sigma^{\text{h,M}_i}_{\text{GB}}, \rho^{\text{h,M}_i}_{\text{GB}}$ | Burst and rate of linear arrival curve of guard band duration |
| $\Delta t_{\text{M}_i}$ | Transmission duration of frames with priority $\text{M}_i$ |
| $\Delta t^{<i}_{\text{AVB}}$ | Duration of higher priority frames from $Q^{<i}_{\text{AVB}}$ |
| $\Delta t_{\text{LP}}$ | Duration of a lower priority frame from $Q^{>i}_{\text{AVB}}$ or $Q_{\text{BE}}$ |
| $\Delta t_{\text{GB}}$ | Guard band duration |
| $\Delta t^{<i}_{\text{OH}}$ | Overhead duration due to higher priority preempted frames |
| $\Delta t^{\text{M}_i}_{\text{OH}}$ | Overhead duration due to preempted frames of priority $\text{M}_i$ |
| $\Delta t_{\text{ST}}$ | Time slots reserved for ST traffic |

$l_{\text{OH}} = 24$ bytes [31] will be added to each fragment of a preempted frame. As described before, the preemption with HOLD/RELEASE mechanism will prevent the ST window from being interfered with by the preemptible traffic while reducing the impact of the GB+ST duration on the link bandwidth that is available to preemptible traffic. Table 1 summarizes the notations used in this paper.

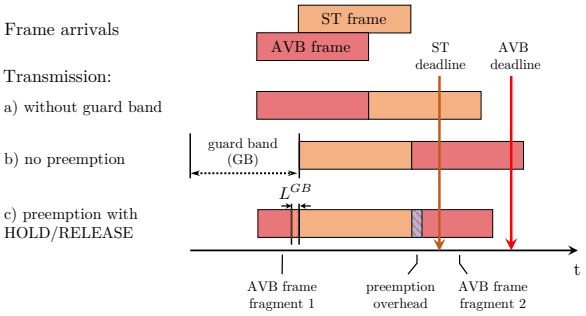


**Figure 2: Frame Transmission in Preemption with HOLD/RELEASE**

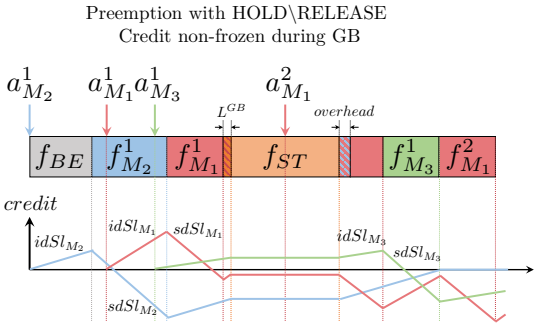


**Figure 3: AVB Frame Transmission in Preemption with HOLD/RELEASE (inspired by [38]).**

## 4.1 Service Curve for AVB Traffic

In [40], a Network Calculus (NC)-based analysis for two AVB classes in the TSN/TAS+CBS architecture has been proposed. However, it only supports the non-preemptive and preemptive without HOLD/RELEASE modes and assumes a frozen credit behavior during the guard band of the non-preemptive mode. A timing analysis for multiple AVB classes that considers both frozen and non-frozen credit behavior for the guard band interval of the non-preemptive mode has been presented in [39]. Neither of the papers discussed the performance analysis in the preemptive with HOLD/RELEASE mode. Although the preemptive mode with HOLD/RELEASE has been discussed in [7], the authors consider the overhead and ST window together, which introduces pessimism since the credit of the AVB traffic class will be reduced/increased rather than frozen during the preemption overhead segment. In this section, we give the service curve to multiple AVB classes $\text{M}_i$ ($i \in [1, n^{\text{h}}_{\text{CBS}}]$), extended from [39] for the preemptive mode with HOLD/RELEASE.

Theorem 1. *The (min-plus) minimum service curve for an AVB Class $M_i$ ($i \in [1, n^{h}_{CBS}]$) in egress port h under preemption with HOLD/RELEASE is*

$$\beta^{h,[PrH/R]}_{M_i}(t) = idSl_{M_i}\left[t - \frac{\alpha^{h}_{ST}(t)}{C} - \frac{\alpha^{h,M_i}_{OH}(t) + c^{\max}_{M_i}}{idSl_{M_i}}\right]^{+}_{\uparrow}. \quad (1)$$

*where $\alpha^h_{ST}(t)$ (c.f. [7, 39, 40]) is the arrival curve of ST traffic scheduled according to the pre-defined GCLs given by Lemma 1, $\alpha^{h,M_i}_{OH}(t)$ is the arrival curve with respect to the extra overheads due to the preemption mode given by Lemma 2, and $c^{\max}_{M_i}$ given by Lemma 3 is the maximum credit bound of Class $M_i$ with non-frozen credit during GB, which is compliant with the standard hypothesis (c.f. Figure 3). The proof can be found in Section 2 of the supplementary material archived in [34].*

For ST traffic scheduled according to the GCL, we follow Lemma 1 of [39], defining the starting time $o^{\mathrm{h}}_i$ and the duration $L^{\mathrm{h}}_{\mathrm{ST},i}$ for each ST window, and the relative offset $o^{\mathrm{h}}_{j,i} = o^{\mathrm{h}}_j - o^{\mathrm{h}}_i$ between the $i^{th}$ and $j^{th}$ ST window instances. The number of ST windows within the GCL period is bounded by $N^{\mathrm{h}}_{\mathrm{ST}}$ due to the schedule cycle $p^{\mathrm{h}}_{\mathrm{GCL}}$ for port $h$.

Lemma 1 ([39]). *The arrival curve of ST traffic in an egress port h is given by, for all $t \in \mathbb{R}^+$*

$$\alpha^h_{ST}(t) = \max_{0\le i\le N^h_{ST}-1}\left\{\alpha^h_{ST,i}(t)\right\}$$
$$\alpha^h_{ST,i}(t) = \sum_{j=i}^{i+N^h_{ST}-1} L^h_{ST,j}C\cdot\left\lfloor\frac{t-o^h_{j,i}}{p^h_{GCL}}\right\rfloor, \tag{2}$$

*where $\alpha^h_{ST,i}(t)$ is one possible arrival curve, computed by taking the $i^{th}$ ST window, $i \in \left[0, N^h_{ST,i}-1\right]$, as the reference; $L^h_{ST,j}\cdot C$ is the maximum amount of bits that can be sent during a ST traffic window of length $L^h_{ST,j}$. As stated in [39], each staircase function $\alpha^h_{ST,i}(t)$ represents the upper bound of ST transmission in the periodic ST windows of length $L^h_{ST,j}$, with $o^h_{j,i}$ being the offsets of the respective ST windows within the GCL period. As mentioned in [39], the proof can be readily derived from the proof for TT streams in TTEthernet [36].*

Since ST traffic is configured as express and other traffic types, i.e., AVB and BE traffic, are configured to be preemptable, the extra overhead of preempted frames can only occur after each ST window [40]. For the preemption without HOLD/RELEASE mode, a preemptable frame can start at any idle time before an ST window and will continue to transmit at most $l^{\max}_{\mathrm{nPr}}+l^+_{IFG}$ bytes after the gate open of ST traffic. Therefore, in the worst case, there will be an overhead after each ST window, as long as two consecutive ST windows are not back to back, as discussed in [40]. However, for the preemption with HOLD/RELEASE mode, there is a guard band with the maximum size of $L_{\mathrm{GB}} = (l^{\max}_{\mathrm{nPr}}+l^+_{IFG})/C$ preserved to prevent the jitter of ST traffic and to prevent any leftover frame segments smaller than $l^{\min}_{\mathrm{Pr}}$ bytes. Therefore, if the maximum frame length of $\mathrm{M}_i$ in the current node port is $l^{\mathrm{h,max}}_{\mathrm{M}_i} \le l^{\max}_{\mathrm{nPr}}$, or if the idle interval time between any two adjacent ST windows is $o^{\mathrm{h}}_{j,i}-o^{\mathrm{h}}_{j-1,i}-L^{\mathrm{h}}_{\mathrm{ST},j-1} \le L_{\mathrm{GB}}$, there will be no preemption overhead after the $j^{th}$ ST window. Thus, for the preemption with HOLD/RELEASE mode, we have

$$l^{\mathrm{h},\mathrm{M}_i}_{\mathrm{OH},j} = l_{\mathrm{OH}}\cdot\mathbb{1}_{\{l^{\mathrm{h,max}}_{\mathrm{M}_i}>l^{\max}_{\mathrm{nPr}}\}}\cdot\mathbb{1}_{\{o^{\mathrm{h}}_{j,i}-o^{\mathrm{h}}_{j-1,i}-L^{\mathrm{h}}_{\mathrm{ST},j-1}>L_{\mathrm{GB}}\}}. \tag{3}$$

Lemma 2. *For the preemption mode with HOLD/RELEASE in an egress port h, the arrival curve of the interval where the credit is frozen due to ST windows, for all $t \in \mathbb{R}^+$, is given by*

$$\alpha^{h,M_i}_{OH}(t) = \max_{0\le i\le N^h_{ST}-1}\left\{\alpha^{h,M_i}_{OH,i}(t)\right\}$$
$$\alpha^{h,M_i}_{OH,i}(t) = \sum_{j=i}^{i+N^h_{ST}-1} l^{h,M_i}_{OH,j}\cdot\left\lfloor\frac{t-o^h_{j,i}-L^h_{ST,j}/C}{p^h_{GCL}}\right\rfloor, \tag{4}$$

*where $\alpha^{h,M_i}_{OH,i}(t)$ is, again, a possible arrival curve derived from a reference $i^{th}$ ST window, $i \in \left[0, N^h_{ST,i}\right]$, and $l^{h,M_i}_{OH,j}$ represents the overhead after the $j^{th}$ ST window, calculated from Eq. (3). The difference between the overhead arrival curve for the preemption with HOLD/RELEASE mode discussed in this paper and for the preemption with no HOLD/RELEASE mode proposed in [40] is the way that the overhead length is calculated.*

Similar to above, besides $L_{\mathrm{GB}}$, the actual maximum guard band before each ST window $j$ is also related to the maximum frame $l^{\mathrm{h,max}}_{\le i}$ of AVB traffic with priority higher than or equal to $\mathrm{M}_i$ in

**Algorithm 1** The maximum times a frame of Class $\mathrm{M}_i$ in the node port h can be preempted

**Input:** GCL, $l^{\mathrm{h,max}}_{\mathrm{M}_i}$
**Output:** $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}}$

**Initialize:** $l^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} \leftarrow l^{\mathrm{h,max}}_{\mathrm{M}_i} - l^+_{FCS}$, $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}} \leftarrow 0$, $\overline{n}^{\mathrm{M}_i,\max}_{\mathrm{Pr}} \leftarrow \left\lfloor\frac{l^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}}-l^{\mathrm{F,min}}_{\mathrm{payload}}}{l^{\mathrm{NF,min}}_{\mathrm{payload}}}\right\rfloor$
**Logic Part:**
**if** $\overline{n}^{\mathrm{M}_i,\max}_{\mathrm{Pr}} = 0$ **then**
  $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}} \leftarrow 0$
**else if** $\overline{n}^{\mathrm{M}_i,\max}_{\mathrm{Pr}} = 1$ **then**
  $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}} \leftarrow 1$
**else**
  $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}} \leftarrow 1$, $\overline{n}^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr,tmp}} \leftarrow 1$
  $\overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} \leftarrow l^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} - l^{\mathrm{NF,min}}_{\mathrm{payload}}$
  **for** benchmark $i \leftarrow 0$ **to** $N^{\mathrm{h}}_{\mathrm{ST},i} - 1$ **do**
    $j \leftarrow i$
    **while** $\overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} + l_{FCS} \ge l^{\min}_{\mathrm{Pr}}$ **do**
      **if** $(o^{\mathrm{h}}_{j+1,j} - L^{\mathrm{h}}_{\mathrm{ST},j} - l_{\mathrm{OH}}/C)\cdot C \ge \overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} + l_{FCS} + l^+_{IFG}$ **then**
        break;
      **else**
        **if** $o^{\mathrm{h}}_{j+1,j} - L^{\mathrm{h}}_{\mathrm{ST},j} - l_{\mathrm{OH}}/C \le L_{\mathrm{GB}}$ **then**
          $j \leftarrow j+1$
        **else**
          **if** $(o^{\mathrm{h}}_{j+1,j} - L^{\mathrm{h}}_{\mathrm{ST},j} - l_{\mathrm{OH}}/C - L_{\mathrm{GB}})\cdot C \le l^{\mathrm{NF,min}}_{\mathrm{payload}} + l_{FCS} + l^+_{IFG}$ **then**
            $\overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} \leftarrow \overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} - l^{\mathrm{NF,min}}_{\mathrm{payload}}$
          **else**
            $\overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{leftover}} \leftarrow \overline{l}^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{leftover}} - (o^{\mathrm{h}}_{j+1,j} - L^{\mathrm{h}}_{\mathrm{ST},j} - l_{\mathrm{OH}}/C - L_{\mathrm{GB}} - l_{FCS}/C - l^+_{IFG}/C)\cdot C$
          **end if**
          $\overline{n}^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr,tmp}} \leftarrow \overline{n}^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr,tmp}} + 1$
          $j \leftarrow j+1$
        **end if**
      **end if**
    **end while**
    $n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}} \leftarrow \max\left\{\overline{n}^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr,tmp}}, n^{\mathrm{h},\mathrm{M}_i}_{\mathrm{Pr}}\right\}$
  **end for**
**end if**

the current node port, and the idle gap between two adjacent ST windows $o^{\mathrm{h}}_{j,i} - o^{\mathrm{h}}_{j-1,i} - L^{\mathrm{h}}_{\mathrm{ST},j-1}$.

$$L^{\mathrm{h,M}_i}_{\mathrm{GB},j} = \min\left\{ L_{\mathrm{GB}}, L_{\mathrm{GB}} - \frac{l^{\mathrm{h,min}}_{\leq i} \cdot \mathbb{1}_{\{l^{\mathrm{h,max}}_{\leq i} \leq l^{\mathrm{max}}_{\mathrm{nPr}}\}}}{C}, \right. \\ \left. o^{\mathrm{h}}_{j,i} - o^{\mathrm{h}}_{j-1,i} - L^{\mathrm{h}}_{\mathrm{ST},j-1} \right\}. \tag{5}$$

Lemma 3. *For the preemption with HOLD/RELEASE mode, the impact of overhead is also reflected in the lower bound of credit compared with the non-preemption mode,*

$$c^{\mathrm{min}}_{M_i} = sdSl_{M_i} \cdot \frac{l^{h,\mathrm{max}}_{M_i}}{C} + n^{h,M_i}_{\mathrm{Pr}} \cdot \left( sdSl_{M_i} \cdot \frac{l_{OH}}{C} \right. \\ \left. + idSl_{M_i} \cdot \frac{l^{\mathrm{max}}_{nPr}}{C} \right) \cdot \mathbb{1}_{\{ idSl_{M_i} \cdot l^{\mathrm{max}}_{nPr} < -sdSl_{M_i} \cdot l_{OH} \}}, \tag{6}$$

*where $idSl_{M_i}$ and $sdSl_{M_i}$ are the idle and send slopes of the AVB class $M_i$, respectively (c.f. Figure 2, and $n^{h,M_i}_{\mathrm{Pr}}$ is the maximum number of preemptions for a single frame of Class $M_i$. We derive a function (described in Algorithm 1) to compute the value of $n^{h,M_i}_{\mathrm{Pr}}$.*

*The upper bound on the credit with non-frozen state during guard band is computed using the expression for the non-preemption mode proposed in [39],*

$$c^{\mathrm{max}}_{M_i} = idSl_{M_i} \cdot \frac{\sum_{j=1}^{i-1} c^{\mathrm{min}}_{M_j} - l^{h,\mathrm{max}}_{>i} - \sigma^{h,M_i}_{GB}}{\sum_{j=1}^{i-1} idSl_{M_j} + \rho^{h,M_i}_{GB} - C}. \tag{7}$$

*by replacing $c^{\mathrm{min}}_{M_j}$ with Eq. (6), and the guard band duration with $L^{h,M_i}_{GB,j}$ in Eq. (5) for constructing linear arrival curve of guard band duration with the burst $\sigma^{h,M_i}_{GB}$ and the rate $\rho^{h,M_i}_{GB}$ [39]. The proof can be found in Section 3 of the supplementary material archived in [34].*

In Algorithm 1, the frame payload $l^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}}$ does not consider the overhead of MAC DA, MAC SA, FCS, etc.. According to the preemptable MAC frame format [12], the minimum payload of the first fragment of a preemptable frame is $l^{\mathrm{F,min}}_{\mathrm{payload}} = 42$ bytes, and all of the leftover fragments have a minimum payload of $l^{\mathrm{NF,min}}_{\mathrm{payload}} = 60$ bytes. Hence, the actual maximum preemption times $n^{\mathrm{h,M}_i}_{\mathrm{Pr}}$ of the single frame of Class $\mathrm{M}_i$ is initialized to 0, of which the final result will be calculated by the algorithm, and $\overline{n}^{\mathrm{M}_i,\mathrm{max}}_{\mathrm{Pr}}$ initialized to $\lfloor l^{\mathrm{h,max}}_{\mathrm{M}_i,\mathrm{payload}} - l^{\mathrm{F,min}}_{\mathrm{payload}} / l^{\mathrm{NF,min}}_{\mathrm{payload}} \rfloor$ represents the maximum possible preemption times of the maximum-length frame of Class $\mathrm{M}_i$ without considering the relative position of ST windows. Lines 7 to 27 handle the case in which the length of the current frame or the remaining frames is sufficient to continue preemption. However, since AVB traffic can only be preempted by ST traffic, whether it can continue to be preempted depends on the idle interval between the respective ST windows. Line 10 indicates that different ST windows in the hyperperiod are respectively used as benchmarks. The ST window benchmark here represents the first window that has a preemptive effect on the AVB frame. Then we obtain the maximum number of preemptions times for a frame of AVB Class $M_i$ under the different ST window benchmarks. Line 12 ensures that the leftover fragment is large enough to be preempted. Line 13 ensures that the current idle time slot is large enough to finish the transmission for the remaining AVB fragment. Thus, the AVB fragment will not be preempted again, and line 14 directly jumps to the end of the while loop. Otherwise, if the current idle time slot is not larger than $L_{\mathrm{GB}}$ (line 16), the AVB fragment will not start to transmit during such a slot. If not, there will be at least $l^{\mathrm{NF,min}}_{\mathrm{payload}}$ bytes in transmission during such a time slot (lines 19 to 22).

Theorem 2. *The standard [14] and novel [1] preemption modes with HOLD/RELEASE have the same min-plus minimum service curve (Eq. (1)) for AVB Class $M_i$ ($i \in [1, n^h_{CBS}]$).*

*Proof*: According to Theorem 1, the cumulative service bits for AVB Class $M_i$ during its busy period are represented by the variation of its credit, by the time duration except for ST windows, and the overhead durations (see Eq. (8) below). For the preemption mode with HOLD/RELEASE, the relation of service times for AVB Class $\mathrm{M}_i$, ST traffic windows occupancy, guard bands and preemption overheads in any interval $\Delta t$ is

$$\Delta t_{\mathrm{M}_i} = \left\lceil (t - s - \Delta t_{\mathrm{ST}}) \mathrm{idSl}_{\mathrm{M}_i} - C \cdot \Delta t^{\mathrm{M}_i}_{\mathrm{OH}} - c_{\mathrm{M}_i}(t) + c_{\mathrm{M}_i}(s) \right\rceil / C. \tag{8}$$

Thus, the credit behavior after each ST window may only affect the credit bounds but does not affect the expression of the service curve for AVB Class $M_i$ in Eq. (1). In the following, we will show that credit bounds will not be affected by the credit behavior after ST windows.

According to Lemma 3, the lower bound of credit for Class $\mathrm{M}_i$ is only related to the maximum frame size $l^{\mathrm{h,max}}_{\mathrm{M}_i}$ in Class $\mathrm{M}_i$ and the maximum number of preemptions $n^{\mathrm{h,M}_i}_{\mathrm{Pr}}$ of a frame of Class $\mathrm{M}_i$. The maximum number of preemptions is only related to the frame size and the idle interval duration between two consecutive ST windows but does not depend on how the credit is behaving after the ST window.

Taking the interval $(s, t]$ defined for the credit upper bound for Class $\mathrm{M}_i$ in Lemma 3, and the characteristics of the lower priority traffic that cannot be preempted by other higher priority AVB traffic but can be preempted by ST traffic, there is at most one frame of lower priority traffic in $Q^{>i}_{\mathrm{AVB}}$ or $Q_{\mathrm{BE}}$ interfering with higher priority AVB traffic. Moreover, the difference between the standard and the novel preemption with HOLD/RELEASE is only related to the credit behavior if the low priority frame is preempted by ST traffic.

Since Eq. (14) from the proof of Lemma 3 which can be found in the supplementary material [34] can be rewritten as,

$$c_{\mathrm{M}_i}(t) - c_{\mathrm{M}_i}(s) \leq \frac{c_{<i}(t) - c_{<i}(s) - (\Delta t_{\mathrm{LP}} + \Delta t_{\mathrm{GB}})}{\sum_{j=1}^{i-1} \mathrm{idSl}_{\mathrm{M}_j} - C} \cdot \mathrm{idSl}_{\mathrm{M}_i}, \tag{9}$$

only the term $\Delta t_{\mathrm{LP}}$ may be influenced by different credit behavior after a ST window. In the standard preemption mode [14], a lower priority frame in $Q^{>i}_{\mathrm{AVB}}$ or $Q_{\mathrm{BE}}$ preempted by the ST traffic will immediately resume its transmission after the ST window. Thus, the maximum interference from the lower priority traffic is equal to $l^{\mathrm{max}}_{>i}$. In the novel preemption mode [1], the lower priority frame preempted by the ST traffic will not be resumed after the ST window if there is a higher priority frame eligible for being forwarded. Therefore, the interference from the lower priority

traffic may be smaller. However, the starting time $s$ may be far away from the ST window so that the lower priority frame will not be preempted. In this case, the maximum interference from the lower priority traffic is still $l^{\max}_{>i}$. Therefore, no matter which preemption mode with HOLD/RELEASE we select, there is always $\Delta t_{\text{LP}} \cdot C \leq \max_{j \in [i+1, n^h_{\text{CBS}}]} \{l^{\max}_{\text{M}_j}, l^{\max}_{\text{BE}}\} = l^{\max}_{>i}$.

Thus, the standard [14] and novel [1] preemption modes with HOLD/RELEASE have the same min-plus minimum service curve for each AVB class. ■

## 5 AVB-AWARE HEURISTIC SCHEDULING ALGORITHM

Using our novel NC analysis for TSN networks with preemption, we now address the problem of generating GCL schedules that ensure the schedulability of both ST and maximize the schedulability of AVB streams. This section presents a heuristic algorithm with a polynomial-time complexity for finding GCLs such that the ST streams meet their deadlines while also aiming for the schedulability of AVB traffic. The routing for AVB traffic is predefined and fixed since the rerouting of both traffic simultaneously would severely increase the complexity of the algorithm. The heuristic algorithm is called AVB-aware Heuristic Scheduling Algorithm (ARIEL), which is based on the heuristic algorithm proposed by Raagaard et al. [27] and Gavriluţ et al. [10]. ARIEL is presented in Algorithm 2.

In the proposed algorithm, we first sort all ST streams using sortFlows($flows_{ST}$) in line 2, based on their periods (or deadlines) in ascending order, since generally, streams with shorter periods (or deadlines) are harder to schedule. We, therefore, aim to incrementally schedule the most "difficult" ST streams first. For arbitrary deadlines, the ordering can be done in deadline-monotonic fashion. After ordering all of the ST streams, we consider each stream individually in a large loop, starting from line 3 in the algorithm.

The main objective of the heuristic is to maximize the number of schedulable AVB streams, and the secondary objective is to minimize the required number of queues for the ST streams while meeting all the timing requirements of the ST streams. Therefore, the algorithm considers all possible stream traffic routes to find a route that provides a higher chance of schedulability for both ST and AVB streams. Therefore, we sort the possible routes of the ST stream on line 4 of the algorithm using flow.sortRoutes(), which is explained in detail in Section 5.2. Then, the stream is set to use only one queue on all links (line 6). After setting the route for the stream that we are scheduling, the algorithm finds a schedule for the stream for all links on its dedicated route in line 15. The scheduleFlow(flow, schedule) function is described in details in Section 5.1. After finding a schedule for the current stream, the algorithm checks whether the AVB streams are schedulable in line 16 using the AVB analysis described in Section 4. If the number of schedulable AVB streams is the highest encountered with the current route (line 17 and onward), the route is set as the best route for the current stream, and the algorithm continues to schedule the next stream. However, if scheduleFlow(flow, schedule) returns false, i.e., there is no schedule found for the current stream, the algorithm checks the problematic link using the constrainingQueue(flow, schedule) function in line 24. This function returns the link on which the offset for the frames was disregarded due to the queue congestion. Therefore, the algorithm increases the number of queues for the ST streams in order to increase the chance of schedulability in line 25 onward.

In some cases, it may be necessary to use more queues for scheduled traffic, and, typically, the number of available ST queues is larger than 1 [6]. The algorithm will increase the number of queues for the current stream under schedule until reaching the maximum number of available ST queues. If the stream cannot be scheduled with the maximum number of queues, the algorithm will use another route to try from the list of sorted possible routes.

Finally, the algorithm sets the best route for the stream and adds it to the schedule. If no best route is set, the stream is not schedulable, and the algorithm returns false since all of ST streams have to be scheduled for a feasible schedule (line 35 onward). Otherwise, if all the streams are scheduled by checking different routes, the algorithm returns the whole schedule in line 41.

The runtime complexity of the algorithm is polynomial since ARIEL only looks at a limited number of (best) routes per each ST flow (AVB flow routing is fixed) and, for each flow, there is no backtracking beyond extending the number of available ST queues up to a maximum of 8. Moreover, the scheduleFlow() function (described below) is a simplified version of the same function from [27] that has been shown to be polynomial.

**Algorithm 2** ARIEL algorithm

**Input:** $flows_{ST}$
**Output:** $schedule$
1: schedule = []
2: $sortedFlows_{ST}$ = sortFlows($flows_{ST}$)
3: **for** flow in $sortedFlows_{ST}$ **do**
4: flow.sortRoutes()
5: success = FALSE
6: flow.setNumQueuesAllLinks(1)
7: **while** !flowRoutes.isEmpty() **do**
8: maxSchedulableAVBs = 0
9: bestSTroute = NULL
10: flowRoutes = flow.getRoute()
11: currentRoute = flowRoutes.pop()
12: flow.setRoute(currentRoute)
13: **while** sucess == FALSE **do**
14: **if** scheduleFlow(flow, schedule) **then**
15: schedule = schedule.add(flow)
16: shedulableAVBs = checkAVB(schedule)
17: **if** shedulableAVBs > maxSchedulableAVBs **then**
18: bestSTroute = currentRoute
19: maxSchedulableAVBs = shedulableAVBs
20: **end if**
21: schedule = schedule.remove(flow)
22: success = TRUE
23: **else**
24: problemLink=constrainingQueue(flow, schedule)
25: newNumQueues=flow.getNumQueues(problemLink)+1
26: flow.setNumQueues(problemLink, newNumQueues)
27: maxNumQueues=flow.getMaxNumQueues(problemLink)
28: **if** newNumQueues > maxNumQueues **then**
29: break
30: **end if**
31: **end if**
32: **end while**
33: **end while**
34: **if** bestSTroute == NULL **then**
35: return FALSE
36: **else**
37: flow.setRoute(bestSTroute)
38: schedule = schedule.add(flow)
39: **end if**
40: **end for**
41: return schedule

## 5.1 scheduleFlow() function

Algorithm 3 presents the process of scheduling a stream over its route. The algorithm is a slightly modified version of an existing scheduling algorithm presented by Raagaard et al. [10, 27], adapted to our problem. The algorithm schedules all frame instances of a given ST stream (flow) from sender to receiver starting at line 3. In the first phase, the algorithm calculates the lower and upper bounds of the offset for the stream in line 4 and line 5, respectively. The lower bound function returns the earliest possible offset in the feasible region for an offset. If the current link is connected to the sender node, the function will return the lower offset bound to be 0; otherwise, the lower bound is the earliest time after adding the sending offset and transmission duration on the previous link and, additionally, the synchronization precision (c.f. [5]). The upper bound function returns the latest point in time where the queue of egress port is available for transmission of the frame on this link.

After finding the offset bound for the stream, the offset on the current link is calculated using middleOffset() function in line 6. In the ASAP strategy of Raagaard et al. [27], the frame is placed at the beginning of the first feasible offset region. In our adaptation, the function tries to schedule the frame in the middle of the first feasible offset region on the link. This middle point was selected to create a gap between the ST frames to schedule AVB frames (similar to the porosity concept from [11, 29]). For example, if we would schedule all ST frames one after the other at the beginning (or the end) of the available offset region (ASAP or ALAP strategies in [27]), the AVB frame might miss its deadline depending on when it is released. An undefined return value from the function means no offset could be found for the frame, and the stream is not schedulable. If the offset is less than or equal to the upper bound, then the algorithm sets the calculated offset as the frame offset on the current link (line 9 and onward). Note that after finding the offset of the stream on the current link, the upper bound offset for the subsequent link should be updated using latestQueueTime() function. However, with this approach of placing the frame in the middle of the region, we eliminate the backtracing of the offset if it is bigger than the upper bound as done in [27] since the offset will never be larger than the upper bound. Therefore, the only condition in line 9 is to get the offset less or equal to the upper bound.

In the end, after all of the frames have been scheduled, the function checks if the end-to-end delay is less than or equal to the period (or deadline) of the stream (line 17). If it is, it returns that the stream is schedulable. Otherwise, the stream is not schedulable.

**Algorithm 3** Schedule Flow

```
Input: flow
Input: schedule
Output: boolean
 1: currentLink = flow.s
 2: frame = flow.getFrame()
 3: while currentLink != flow.t do
 4:     lowBound = calcLowOffset(flow, currentLink, frame, schedule)
 5:     upBound = calcUpOffset(flow, currentLink, frame, schedule)
 6:     offset = middleOffset(flow, frame, lowBound, upBound)
 7:     if offset == indefinite then
 8:         return false
 9:     else
10:         flow.setOffset(currentLink, frame, offest)
11:         nextLink = flow.getNextLink(currentLink)
12:         upBound = latestQueueTime(nextLink, schedule, frame, offset)
13:         flow.setUpOffsetBound(nextLink, frame, upBound)
14:         currentLink = nextLink
15:     end if
16: end while
17: return (flow.end2end <= flow.deadline (≤ flow.period))
```

## 5.2 ST route ordering

In this subsection, we present flow.sortRoutes() function in Algorithm 2. There are multiple ways to find a route for a stream, including the shortest route based on the number of links or the route with less accrued link utilization. However, using only the number of hops (e.g. [10]) as a metric to sort/select the list of possible routes may not always lead to a feasible schedule [17]. Other works, e.g. [17, 26] use the link utilization or the load balancing [24] as an alternative metric. However, we introduce a novel approach to route finding in heuristic TSN approaches where we consider several different properties that affect schedulability combined into one sorting metric. We first find all k-possible routes for a stream according to the k-shortest path method, i.e., according to the number of hops on the stream's route. Afterward, we use the following metric to reorder the routes to increase the schedulability probability of both ST and AVB streams and the probability of reaching a feasible solution more quickly.

First, we consider the number of links ($N_r$) on route $r$ as with previous approaches. Clearly, the more links on the route, the more message instances we will have to schedule overall, leading to a higher overall network load. We normalize the number of links to $n_r = \frac{N_r - N_{min}}{N_{max} - N_{min}}$, where $N_{min}$ and $N_{max}$ are the minimum and maximum number of links from the talker to the listener nodes, respectively. Moreover, we use similar to [17] the link utilization, denoted by $u^h$, on link $h$ of the route $r$. When the link is less utilized, there is a higher chance for a new ST frame that can be scheduled on that link. Afterwards, we define the average utilization as $\overline{u}_r = \sum_{h \in r}\{u^h\}$ on the route $r$. The standard deviation of the utilization can be computed as $\hat{u}_r = \sqrt{\frac{1}{N_r}\sum_{h \in r}(u^h - \overline{u})^2}$ on the route $r$. The high value for the standard deviation shows the more imbalance distribution of ST streams over the route; hence there is a higher chance for the route to reach a bottleneck in the scheduling of ST streams. We also need to consider the number of AVB streams, denoted by $N^h_{AVB}$, that is going through link $h$ on the route $r$. The normalized metric is $n^h_{AVB} = N^h_{AVB}/N^{total}_{AVB}$, where $N^{total}_{AVB}$ is the total number of AVB streams in the network. Given the above metrics, we can combine them to obtain a weighted sum in order to reorder the possible routes accordingly. The combined metric for reordering the routes of a stream can be expressed as

$$M = w_1 \cdot n_r + w_2 \cdot \overline{u}_r + w_3 \cdot \hat{u}_r + w_4 \cdot n^h_{AVB}. \tag{10}$$

The combined metric is used for ordering the ST stream routes with lower values of $M$ being more favorable over higher values in terms of the probability of being able to schedule the respective ST stream. Naturally, the weights in the metric have to be chosen carefully for the given use-case or domain, and we note that there is no silver bullet here. Usually, the weights are uniformly chosen to give all metrics the same degree of influence, or they are chosen unequally to reflect preferences for, e.g., shorter paths or more balanced link utilization.

# 6 EXPERIMENTAL EVALUATION

In this section, we show the performance of the proposed scheduling algorithm with respect to the schedulability of AVB traffic and the algorithm runtime with and without preemption enabled. Moreover, we show the impact on AVB schedulability with different link utilization configurations. We chose equal weights for the route ordering metric described in Section 5.2. The first experiment is based on a real-world use case, while the second one uses synthetic traffic. ARIEL was implemented in Python running on a computer with an Intel Core i7-10750H CPU running at 2.60 GHz and featuring 32 GB of RAM.

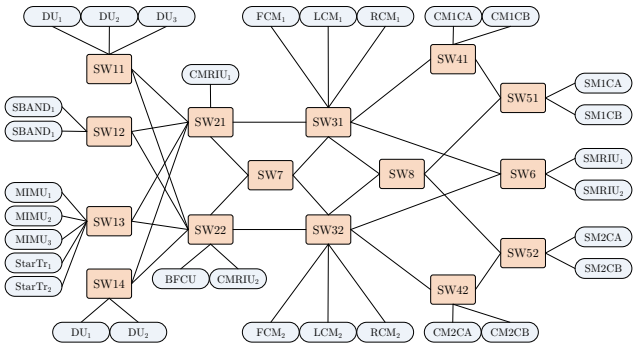


**Figure 4: Network topology of Orion use case (c.f [35]).**

## 6.1 Realistic use-case experiment

For the real-world use case study, we consider the Orion Crew Exploration Vehicle (CEV) presented in [25] and used in several other works (c.f. [30, 35, 37]). Since the original use-case featured a relatively low link utilization, we kept the topology and base stream definition but extended the use-case by adding additional communication streams. The network in this use case contains 31 end systems connected via 15 TSN switches, as shown in Figure 4. The network initially contained 100 ST and 33 AVB streams, which we extended to 222 ST and 149 AVB streams to increase the utilization and better showcase the difference between the preemptive and non-preemptive modes. The stream size is selected between 135 and 1527 bytes, and the period is within the range [7.5, 375]ms. The use case predefines the routes of the streams and the deadlines are considered implicit, i.e., they are equal to the periods. Please note that our algorithm is able to also work with arbitrary deadlines, but the sorting of the ST flows would have to be modified to reflect a deadline-monotonic ordering.

**Table 2: Schedulability of the streams under different modes**

| Mode | ST streams | AVB streams |
|---|---|---|
| Non-preemption | 222/222 (100%) | 128/149 (85,9%) |
| Preemption | 222/222 (100%) | 149/149 (100%) |

We used ARIEL to schedule the ST streams in the use case while considering the schedulability of the AVB streams in two modes of enabled and disabled frame preemption. The results are shown in Table 2. We can see from the results that all of the ST and AVB traffic are schedulable when preemption is enabled, while in the non-preemption mode, only 128 AVB streams out of the 149 (85, 9%) are schedulable. The runtime of ARIEL was 7090.13$s$ without preemption and 602.5$s$ with preemption. The relatively high difference in runtime can be explained by the runtime of the NC analysis calls, which, for this use case, was higher in the non-preemptive mode compared to the preemptive mode.

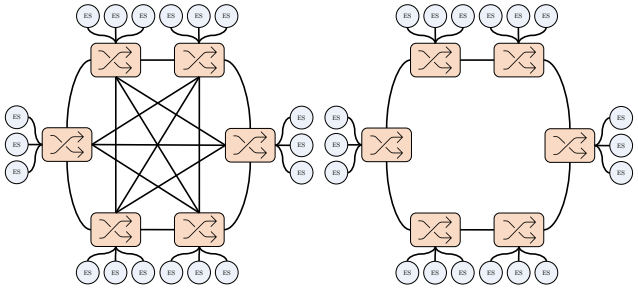

**Figure 5: Network topologies used for synthetic tests**

## 6.2 Synthetic use-case experiments

We implemented a traffic generation tool that generates synthetic traffic for two networks topology types, mesh and ring, as depicted in Figure 5. The network topologies are further subdivided into medium-sized networks, containing 4 switches and 13 end-systems, and large networks with 8 switches and 48 end-systems, as proposed in [4]. Therefore, we use four network topologies to evaluate the proposed scheduling algorithm, i.e., medium mesh, medium ring, large mesh, and large ring topologies. The generator randomly selects a stream period from the set $\{1, 2, 5, 10\}$ms, according to the use case defined in [16], and sets the deadlines to be equal to the periods. The payload of all generated streams is 1500 bytes, and the network speed was set to 100 Mbps.

We have generated 100 sets of traffic per network scenario and scheduled them using ARIEL with and without preemption. Since it is difficult to generate the exact utilization on all network links uniformly, especially since only the routes of AVB streams are fixed at the design time, the generator creates an equal number of ST and AVB streams until at least one of the links reaches the desired utilization when only considering the routed AVB streams. Hence, after routing scheduling the ST streams, the combined ST+AVB utilization on some links will be higher than the set utilization. While we generate test cases where all ST streams are schedulable, not all AVB streams may be schedulable depending on, e.g., the utilization of the network. We selected the desired peak AVB utilization to be 60% for the first set of test cases since, usually for TSN networks, around 25% of the bandwidth is reserved for best-effort traffic, leaving around 75% for time-sensitive communication [15, 21]. Hence, when generating use-cases with peak 60% AVB utilization, the combined ST+AVB utilization typically exceeds the threshold value of 75%. We have also looked at test cases with 50%, 60%, 70%, and 80% peak AVB utilization in large ring network topologies in order to see the effect of the link utilization on AVB and ST schedulability.

| | M RING P | M RING NP | M MESH P | M MESH NP | L RING P | L RING NP | L MESH P | L MESH NP |
|---|---|---|---|---|---|---|---|---|
| Average | 34,50 | 25,22 | 32,66 | 23,09 | 34,80 | 25,75 | 49,97 | 46,56 |
| Median | 32,58 | 24,22 | 30,64 | 23,72 | 34,88 | 24,52 | 48,63 | 46,58 |
| StDev | 12,30 | 12,72 | 10,98 | 10,32 | 13,48 | 12,43 | 11,59 | 11,59 |

**Table 3: AVB schedulability [%]: average, median, and standard deviation for medium and large (M, L) mesh and ring (MESH, RING) topologies with preemption (P) and without preemption (NP).**

| | M RING P | M RING NP | M MESH P | M MESH NP | L RING P | L RING NP | L MESH P | L MESH NP |
|---|---|---|---|---|---|---|---|---|
| Average | 78,28 | 75,30 | 156,27 | 183,52 | 100,35 | 99,07 | 276,47 | 269,98 |
| Median | 71,72 | 66,51 | 136,61 | 174,23 | 86,29 | 83,97 | 190,57 | 199,62 |
| StDev | 48,72 | 46,10 | 110,60 | 97,05 | 59,25 | 59,02 | 500,47 | 490,66 |

**Table 4: Runtime[sec]: average, median, and standard deviation for medium and large (M, L) mesh and ring (MESH, RING) topologies with preemption (P) and without preemption (NP).**

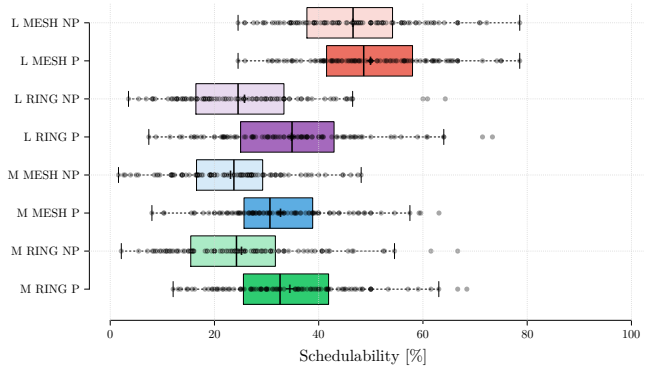


**Figure 6: AVB schedulability results for medium and large (M, L) mesh and ring (MESH, RING) topologies with preemption (P) and without preemption (NP).**

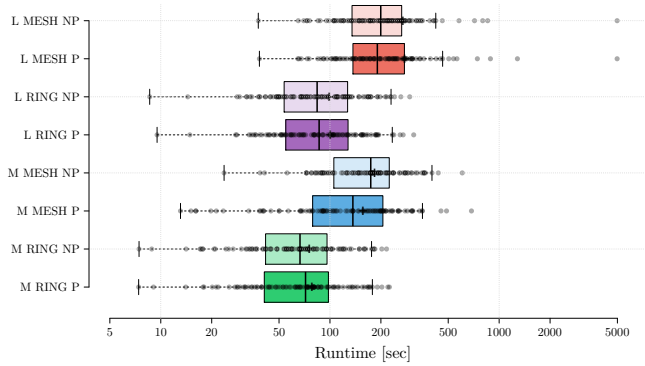


**Figure 7: Runtime results for medium and large (M, L) mesh and ring (MESH, RING) topologies with preemption (P) and without preemption (NP).**

The numerical results of the evaluation can be found in Table 3 and Table 4, where we showcase the average, median, and standard deviation of the results for AVB schedulability and runtime per network scenario, respectively. Furthermore, Figures 6 and 7, depict the AVB schedulability and algorithm runtime, respectively, as box plots, also showing the individual use-case data points to better visualize the spread and distribution of the results. The lower and upper whisker boundaries are set to show low outliers that are below $1,5 \times IQR$ of the $1^{st}$ quartile and high outliers that are over $1,5 \times IQR$ of the $3^{rd}$ quartile. In Figure 6, the x-axis represents the percentage of AVB streams that could be scheduled such that the respective deadlines were fulfilled, while in Figure 7, the logarithmic x-axis represents the total runtime in seconds for scheduling both ST and AVB streams.

In all cases, ARIEL managed to schedule all ST streams, this being the primary goal of the algorithm, while AVB streams were not all schedulable. On average, we have around 10% higher AVB schedulability when the preemption is enabled compared to when it is disabled (c.f. Table 3). The only exception is for the experiment on the large mesh network in which we observe only 3.4% improvement in the schedulability. Mesh networks typically take more time to schedule since there are a larger number of links to schedule and, therefore, a more significant number of routes for ST traffic to select from. Moreover, having more route options in large networks usually also leads to an increased schedulability compared to, e.g., ring topologies of the same size.

In terms of the runtime for each network scenario, we can observe that there is no significant difference between the enabled and disabled preemption modes. The average number of ST+AVB streams was 78, 123, 85, 83, for the medium ring, medium mesh, large ring, and large mesh test cases with 60% utilization, respectively. The increased number of streams explains the increased runtime of the medium mesh (M MESH P & NP in Figure 7) in comparison to the other cases. As a reference, for an exact method like [6], scheduling 100 ST streams in a relatively small network with 7 end-systems and 5 switches takes between 1, 5 minutes to 4 hours depending on the stream periods while not being able to consider the schedulability of potential AVB streams.

There are heuristic methods that are optimized toward scalability (e.g. [32]) that can schedule networks with up to 2000 nodes and $10K$ ST streams in 1 hour, but they do not compute the worst-case AVB bounds in order to check the AVB schedulability within the scheduling step. Other heuristics that schedule ST and AVB streams

together but without preemption take, e.g., 537*s* for 61 streams in a network with 32 end-systems and 18 switches [10]. However, we note that it is not easy to compare the runtime of heuristics in a systematic manner; hence, we primarily emphasize that the benefit of our work is that an increase in AVB schedulability does not come with a significant increase in algorithm runtime when compared to non-preemptive approaches.

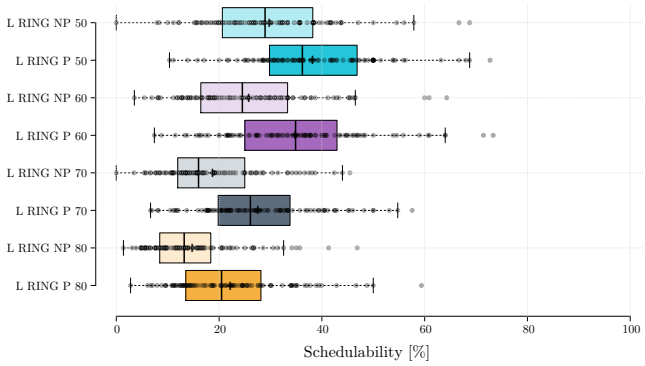


**Figure 8: AVB schedulability results for large (L) ring (RING) topologies with 50%, 60%, 70%, and 80% peak AVB utilization in preemptive (P) and non-preemptive (NP) modes.**

In Figure 8 we show the schedulability of AVB streams when the peak AVB utilization in the large ring topology (L RING) is 50%, 60%, 70%, and 80% both in the preemptive and non-preemptive modes. While the AVB schedulability decreases with increasing peak utilization, we observe that the relative difference in AVB schedulability when comparing the preemptive and non-preemptive modes is similar, with the preemptive mode ensuring on average 8, 3% more AVB schedulability than the non-preemptive mode, independently of the peak utilization.

## 7 CONCLUSION

In this paper, we have studied the creation of GCL schedules considering the integration of ST and AVB traffic in TSN networks with preemption. We have derived new worst-case response time (WCRT) bounds based on Network-Calculus for an arbitrary number of AVB classes and different configurations for the CBS credit behavior. Using this new WCRT result, we have presented a heuristic algorithm for scheduling ST traffic while taking the schedulability of the lower priority AVB streams into account. We evaluated our approach using both an experiment derived from a real-world use-case as well as using synthetic workloads in (medium and large) mesh and ring topologies. We have shown the performance of our approach both in terms of runtime and in terms of increasing the AVB schedulability when preemption is enabled.

In future work, we want to investigate whether allowing preemption within the ST traffic classes leads to an improvement in schedulability. Moreover, we aim to study more advanced heuristic methods that integrate the network calculus analysis more closely into the ST scheduling loop and add optimization objectives in order to optimize the latency of both ST and AVB streams.

## REFERENCES


[1] Mohammad Ashjaei, Mikael Sjödin, and Saad Mubeen. 2021. A Novel Frame Preemption Model in TSN Networks. *Journal of Systems Architecture* 80 (June 2021), 1–30.

[2] D. Bruckner, R. Blair, M-P. Stanica, A. Ademaj, W. Skeffington, D. Kutscher, S. Schriegel, R. Wilmes, K. Wachswender, L. Leurs, M. Seewald, R. Hummen, E-C. Liu, and S. Ravikumar. 2018. OPC UA TSN A new Solution for Industrial Communication. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/opc-ua-tsn-solution-industrial-comm-wp.pdf.

[3] Martin Böhm and Diederich Wermser. 2021. Multi-Domain Time-Sensitive Networks—Control Plane Mechanisms for Dynamic Inter-Domain Stream Configuration. *Electronics* 10, 20 (2021). https://doi.org/10.3390/electronics10202477

[4] Silviu S. Craciunas and Ramon Serna Oliver. 2016. Combined Task- and Network-level Scheduling for Distributed Time-Triggered Systems. *Journal of Real-Time Systems* 52, 2 (2016), 161–200.

[5] Silviu S. Craciunas and Ramon Serna Oliver. 2021. Out-of-sync Schedule Robustness for Time-sensitive Networks. In *Proc. WFCS*. IEEE.

[6] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelik, and Wilfried Steiner. 2016. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proc. RTNS*. ACM.

[7] Hugo Daigmorte, Marc Boyer, and Luxi Zhao. 2018. Modelling in network calculus a TSN architecture mixing Time-Triggered, Credit Based Shaper and Best-Effort queues. *Technical report* (2018).

[8] Jonathan Falk, Frank Dürr, and Kurt Rothermel. 2018. Exploring Practical Limitations of Joint Routing and Scheduling for TSN with ILP. In *Proc. RTCSA*.

[9] Anais Finzi and Silviu S. Craciunas. 2019. Integration of SMT-based Scheduling with RC Network Calculus Analysis in TTEthernet Networks. In *Proc. ETFA*. IEEE.

[10] Voica Gavriluț, Luxi Zhao, Michael L. Raagaard, and Paul Pop. 2018. AVB-Aware Routing and Scheduling of Time-Triggered Traffic for TSN. *IEEE Access* 6 (2018), 75229–75243. https://doi.org/10.1109/ACCESS.2018.2883644

[11] Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, and Saad Mubeen. 2021. Synthesising Schedules to Improve QoS of Best-effort Traffic in TSN Networks. In *Proc. RTNS*. ACM.

[12] IEEE. 2015. P802.3br — Standard for Ethernet Amendment Specification and Management Parameters for Interspersing Express Traffic. *Amendment to IEEE Std 802* (2015).

[13] IEEE. 2016. Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/tsn.html. retrieved 23.09.2021.

[14] IEEE. 2018. 802.1Q—IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks. https://standards.ieee.org/standard/802_1Q-2018.html.

[15] Jaewoong Ko, Ju-ho Lee, Chulsun Park, and Sung-kwon Park. 2015. Research on optimal bandwidth allocation for the scheduled traffic in IEEE 802.1 AVB. In *Proc. ICVES*. 31–35. https://doi.org/10.1109/ICVES.2015.7396889

[16] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. 2015. Real world automotive benchmarks for free. In *Proc. WATERS*.

[17] Sune Mølgaard Laursen, Paul Pop, and Wilfried Steiner. 2016. Routing Optimization of AVB Streams in TSN Networks. *SIGBED Rev.* 13, 4 (nov 2016), 43–48. https://doi.org/10.1145/3015037.3015044

[18] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag.

[19] Lucia Lo Bello, Mohammad Ashjaei, Gaetano Patti, and Moris Behnam. 2020. Schedulability analysis of Time-Sensitive Networks with scheduled traffic and preemption support. *Journal of Parallel and Distributed Computing*, 144, (2020).

[20] Rouhollah Mahfouzi, Amir Aminifar, Soheil Samii, Ahmed Rezine, Petru Eles, and Zebo Peng. 2018. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In *Proc. DATE*.

[21] Ahmed Nasrallah, Akhilesh S. Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. 2019. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Communications Surveys Tutorials* 21, 1 (2019), 88–145. https://doi.org/10.1109/COMST.2018.2869350

[22] Naresh Ganesh Nayak, Frank Dürr, and Kurt Rothermel. 2018. Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks. *IEEE Trans Industr Inform* 14, 5 (2018).

[23] Maryam Pahlevan and Roman Obermaisser. 2018. Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks. In *Proc. ETFA*. https://doi.org/10.1109/ETFA.2018.8502515

[24] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. 2019. Heuristic List Scheduler for Time Triggered Traffic in Time Sensitive Networks. *SIGBED Rev.* 16, 1 (2019), 15–20.

[25] Michael Paulitsch, E Schmidt, B Gstöttenbauer, C Scherrer, and H Kantz. 2011. Time-triggered communication (industrial applications). *Time-Triggered Communication* (2011), 121–152.

[26] Paul Pop, Michael Lander Raagaard, Silviu S. Craciunas, and Wilfried Steiner. 2016. Design Optimization of Cyber-Physical Distributed Systems using IEEE Time-sensitive Networks (TSN). *IET Cyber-Physical Systems: Theory and Applications* 1, 1 (2016), 86–94.

[27] Michael Lander Raagaard and Paul Pop. 2017. Optimization algorithms for the scheduling of IEEE 802.1 Time-Sensitive Networking (TSN). *Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep* (2017).

[28] Ramon Serna Oliver, Silviu S. Craciunas, and Wilfried Steiner. 2018. IEEE 802.1Qbv Gate Control List Synthesis using Array Theory Encoding. In *Proc. RTAS*. IEEE.

[29] Wilfried Steiner. 2011. Synthesis of Static Communication Schedules for Mixed-Criticality Systems. In *Proc. ISORCW*. https://doi.org/10.1109/ISORCW.2011.12

[30] Domiţian Tămaş-Selicean, Paul Pop, and Jan Madsen. 2014. Design of mixed-criticality applications on distributed real-time systems. *Technical University of Denmark* (2014).

[31] Daniel Thiele and Rolf Ernst. 2016. Formal worst-case performance analysis of time-sensitive ethernet with frame preemption. In *Proc. ETFA*.

[32] Marek Vlk, Kateřina Brejchová, Zdeněk Hanzálek, and Siyu Tang. 2022. Large-scale periodic scheduling in time-sensitive networks. *Computers & Operations Research* 137 (2022), 105512. https://doi.org/10.1016/j.cor.2021.105512

[33] Marek Vlk, Zdeněk Hanzálek, and Siyu Tang. 2021. Constraint programming approaches to joint routing and scheduling in time-sensitive networks. *Computers & Industrial Engineering* 157 (2021), 107317. https://doi.org/10.1016/j.cie.2021.107317

[34] Luxi Zhao, Aldin Berisa, Silviu S. Craciunas, Mohammad Ashjaei, Saad Mubeen, Masoud Daneshtalab, and Mikael Sjödin. 2022. *AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption - Supplementary Material.* https://doi.org/10.5281/zenodo.6190143 Available at https://zenodo.org/record/6190143..

[35] Luxi Zhao, Paul Pop, and Silviu S. Craciunas. 2018. Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus. *IEEE Access* 6 (2018), 41803–41815. https://doi.org/10.1109/ACCESS.2018.2858767

[36] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. 2017. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems,* 52(2), (2017).

[37] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. 2017. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Journal of Real-Time Systems* 53, 2 (2017), 254–287.

[38] Luxi Zhao, Paul Pop, and Sebastian Steinhorst. 2017. Quantitative Performance Comparison of Various Traffic Shapers in Time-Sensitive Networking. *CoRR* abs/2103.13424 (2017). arXiv:2103.13424 https://arxiv.org/abs/2103.13424.

[39] Luxi Zhao, Paul Pop, Zhong Zheng, Hugo Daigmorte, and Marc Boyer. 2021. Latency Analysis of Multiple Classes of AVB Traffic in TSN With Standard Credit Behavior Using Network Calculus. *IEEE Transactions on Industrial Electronics* 68, 10 (2021), 10291–10302. https://doi.org/10.1109/TIE.2020.3021638

[40] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. 2018. Timing analysis of AVB traffic in TSN networks using network calculus. In *Proc. RTAS*.

[41] Yuanbin Zhou, Soheil Samii, Petru Eles, and Zebo Peng. 2021. ASIL-Decomposition Based Routing and Scheduling in Safety-Critical Time-Sensitive Networking. In *Proc. RTAS*. https://doi.org/10.1109/RTAS52030.2021.00023

[42] Yuanbin Zhou, Soheil Samii, Petru Eles, and Zebo Peng. 2021. Reliability-Aware Scheduling and Routing for Messages in Time-Sensitive Networking. *ACM Trans. Embed. Comput. Syst.* 20, 5, Article 41 (2021), 24 pages. https://doi.org/10.1145/3458768

[43] Yuanbin Zhou, Soheil Samii, Petru Eles, and Zebo Peng. 2022. Time-Triggered Scheduling for Time-Sensitive Networking with Preemption. In *Proc. ASP-DAC*. https://doi.org/10.1109/ASP-DAC52403.2022.9712545