Formal Worst-Case Timing Analysis of Ethernet TSN's Time-Aware and Peristaltic Shapers

Daniel Thiele*, Rolf Ernst*, and Jonas Diemer†
*Institute of Computer and Network Engineering, Technische Universität Braunschweig, 38106 Braunschweig, Germany
{thiele,ernst}@ida.ing.tu-bs.de
†Symtavision GmbH, Frankfurter Straße 3c, 38122 Braunschweig, Germany
diemer@symtavision.com

Abstract—**Ethernet is considered as a future communication standard for distributed embedded systems in the automotive and industrial domains. A key challenge is the deterministic low-latency transport of Ethernet frames, as many safety-critical real-time applications in these domains have tight timing requirements. Time-sensitive networking (TSN) is an upcoming set of Ethernet standards, which (among other things) address these requirements by specifying new quality of service mechanisms in the form of different traffic shapers. In this paper, we consider TSN's time-aware and peristaltic shapers and evaluate whether these shapers are able to fulfill these strict timing requirements. We present a formal timing analysis, which is a key requirement for the adoption of Ethernet in safety-critical real-time systems, to derive worst-case latency bounds for each shaper. We use a realistic automotive Ethernet setup to compare these shapers to each other and against Ethernet following IEEE 802.1Q.**

I. Introduction

Packet-switched Ethernet will be used in next-generation automotive communication architectures, as traditional buses such as CAN or FlexRay cannot keep pace with the increasing bandwidth and scalability requirements of advanced driver assistance and infotainment systems. As a switched network, Ethernet provides a scalable, high-speed, and cost-effective communication platform, which allows arbitrary topologies. However, as each switch output port is a point of arbitration, Ethernet exhibits a complex timing behavior, which must be verified thoroughly before it can be used in timing- and safety-critical systems. Ethernet is anticipated to serve as an in-vehicle communication backbone, where is must be able to transport traffic streams of mixed-criticality. This requires quality of service (QoS) mechanisms, in order to provide deterministic timing guarantees to critical traffic.

Standard Ethernet following IEEE 802.1Q introduced 8 traffic classes. These classes can be used to prioritize traffic, which is typically implemented by a static-priority non-preemptive (SPNP) scheduler at each output port in each switch and endpoint. This limited number of classes requires that multiple traffic streams share a class. Traffic within a shared class is usually scheduled in FIFO order. Ethernet AVB [1] introduced standardized traffic shaping by using a credit-based shaper (CBS) to ensure that shaped traffic classes do not exceed their preconfigured bandwidth bounds. This CBS, however, only shapes the highest priorities, i.e. it introduces additional shaping delays to the most critical ones, which is undesirable for latency-sensitive traffic. Currently, a new set of Ethernet QoS mechanisms is being evaluated for standardization under the name of time-sensitive networking (TSN) [2]. One of TSN's goals is the development of new traffic shapers, which offer tight and deterministic end-to-end latencies for real-time traffic. At this time, TSN's most prominent shapers are: the time-aware, the burst-limiting, and the peristaltic shaper.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080.

There are multiple means to evaluate network performance. Simulation (e.g. [3]) is used to *assess* a network's response to a given set of stimuli. It provides valuable insights into the network's behavior, e.g. path latency histograms. However, simulation often requires long simulation runs and usually does not expose all corner cases (it rather provides a lower bound on the actual worst-case behavior), rendering it unsuitable for the timing *verification* of timing- and safety-critical systems under worst-case conditions. Formal performance analyses like compositional performance analysis (CPA) [4] or real-time calculus (RTC) [5], in contrast, have been proven to efficiently provide safe worst-case (latency) bounds (i.e. upper bounds on the actual worst-case) for Ethernet. Formal analysis methods are becoming even more important with the advent of highly automated and autonomous driving. Both, simulation and formal analysis are important to evaluate automotive Ethernet networks. In this paper, we focus on the latter. A discussion about the gap between the observed latencies from a simulation and the latency guarantees from a formal CPA analysis in the Ethernet context can be found in [6].

The **contribution of this paper** is a set of formal timing analysis methods for TSN's proposed time-aware (TAS) and peristaltic traffic shapers (PS). In contrast to related work [7], we consider all blocking effects, especially those of same-priority traffic streams, which is important given Ethernet's limited number of priority levels. We evaluate the worst-case performance of TAS and PS against each other and against IEEE 802.1Q using a realistic automotive Ethernet setup. Due to space constrains, we do not discuss TSN's burst-limiting shaper. This shaper, however, is quite similar to AVB's CBS, except that it allows limited-sized bursts of critical traffic to pass without prior blocking by lower priority traffic.

II. Related Work

There is a large body of related work on the formal timing analysis of switched Ethernet. An analysis for IEEE 802.1Q, for example, is presented in [8]. Recent improvements of this analysis show that sub-millisecond worst-case latency guarantees are possible for typical automotive setups [9].

Different Ethernet QoS mechanisms have been proposed. The most prominent ones are TTEthernet [10], which provides time-triggered link access for critical traffic, and Ethernet AVB [1], which provides bandwidth-limited event-triggered link access. Formal analyses for Ethernet AVB are presented in e.g. [11], [12]. Both publications show that, in a worst-case analysis, the latency guarantees of high-priority traffic in Ethernet AVB suffer mainly (and severely) from AVB's traffic shapers. [11] further shows that increasing the bandwidth reservation of a shaped AVB traffic class beyond its actual bandwidth requirement (overreservation) helps to achieve shorter latency guarantees. At the maximum overreservation, AVB's latency guarantees match those of IEEE 802.1Q.

A first attempt of a formal timing analysis of TSN's traffic shapers (including the burst-limiting shaper) has been presented in [7]. This analysis, however, has two main limitations: (1) Interference by same-priority frames, i.e. frames of equal priority competing for link access, is not considered. This is a major drawback, as the limited number of Ethernet priorities will require priority sharing in realistic setups and, hence, will introduce same-priority interference. (2) Higher-priority interference impacting PS traffic is not considered, i.e. it is only possible to analyze a single PS traffic class, which must be on the highest priority. Actual systems, however, might have more than one PS traffic class, which might not be on the highest priority.

In contrast, we will present a formal analysis based on the proven CPA framework, which does not suffer from these limitations and, hence, allows the formal analysis of all traffic streams across all traffic classes, including same-priority interference and multiple PS traffic classes.

III. Compositional Performance Analysis of Ethernet

Throughout this paper, we use compositional performance analysis (CPA) [4] to formally derive upper bounds on the worst-case end-to-end latencies of Ethernet frames. In CPA, systems are modeled by three components. *Resources* abstract the actual processing resources and provide service according to a scheduling policy, e.g. SPNP. *Tasks* are mapped to these resources and consume service. The amount of service consumed by a task i per activation can vary between a lower and an upper bound (C_i^- and C_i^+). Distributed systems are modeled as a directed graph, in which tasks correspond to nodes and task dependencies correspond to edges. Whenever a task finishes executing it sends an event to its dependent tasks. Tasks without predecessors must be stimulated by events from external sources. *Event models* are used to abstract task activations. They are defined as a set of event arrival functions $\eta_i^-(\Delta t)$ and $\eta_i^+(\Delta t)$, which describe the lower and upper bound on the number of task activations in any half-open time interval $[t,t+\Delta t)$ [4]. Hence, in contrast to a single event trace, event models capture all possible task activation scenarios within their bounds. Event models can also be defined by the pseudo-inverse of their arrival functions, $\delta_i^+(n)$ and $\delta_i^-(n)$, which describe the maximum and minimum time interval between the first and the last event in any sequence of n events.

CPA is based on an iterative approach. Each resource is analyzed by a *local analysis*, which utilizes the busy period approach [13], to derive new output event models for each task. This is done by constructing a critical instant scenario, which maximizes the response time of the task under analysis by spanning the busy period using worst-case activation (and execution) sequences of all interfering tasks. New output event models can be derived from the response time jitter (maximum minus minimum response time) [4]. The output event model of a task becomes the input event model of its dependent tasks. A *global analysis loop* manages this event model propagation. If all event models become stable, i.e. do not change anymore, the analysis finishes. Otherwise, i.e. if a predefined number of iterations has been reached or some constraint (e.g. deadline, jitter, latency) is violated, the system is deemed unscheduable.

Apart from response times and latencies, CPA is also able to formally derive jitter and buffer size bounds.

It has been shown that Ethernet networks can be mapped to CPA models [14]. The output ports of Ethernet switches are the points of arbitration and are, hence, modeled as resources. Task activations and their executions on a resource correspond to Ethernet frame arrivals and transmissions on an switch port. The transmission times of Ethernet frames (C_i^- and C_i^+) correspond to task execution times and are defined to be the best- and worst-case frame transmission times on a switch port in the absence of any interference. For the frames of a traffic stream i, these transmission times can be defined as

$$C_i^{+/-} = \frac{42\text{bytes} + \max\{42\text{bytes}, p_i^{+/-}\}}{r_{TX}} \tag{1}$$

where $p_i^{+/-}$ is the stream's maximum/minimum payload size and r_{TX} is the port's transmission rate in bytes/second. The constant terms account for minimum Ethernet frame size, protocol overhead, IEEE 802.1Q tagging, and inter-frame gap.

Finally, a traffic stream through an Ethernet network, i.e. a sequence of Ethernet frames between a source and one (or multiple) destination(s), which receive identical QoS policies, is modeled as a chain of tasks, where the edges between tasks represent their communication dependencies. The frames of a traffic stream are transmitted along this task chain.

In the following sections, we present local analyses for different Ethernet schedulers and shapers, which can be used in CPA. We start with a brief overview of an IEEE 802.1Q analysis, which is largely based on related work [12], in order to introduce a baseline analysis and to establish a common notation. In subsequent sections, where we present analyses of TSN shapers, we reference and extend this baseline analysis.

IV. Standard Ethernet (IEEE 802.1Q)

The goal of the local analysis is to derive the worst-case transmission latency of a frame at a particular switch. The transmission latency of a frame is the time from when it has been received at a switch's input port until it has been fully transmitted from an output port. Inside a switch the transmission of a frame is affected by several delays: queueing delay at the input port, forwarding delay in the switch fabric, queueing delay at the output port, and transmission delay on the link. The first two delays depend on the switch implementation and are typically in the order of a few clock cycles, i.e. they are orders of magnitude apart from the actual transmission latencies. The transmission delay on the link can be considered constant and is typically in the order of nanoseconds. Hence, we assume that the transmission latency of a frame comprises only the output port's queueing delay, as the other delays only have negligible impact on the transmission latency (or can be added as constants to the other delays). The output port's queueing delay accounts for all delays induced by an output port's scheduler (including the various delays and shaper effects of TSN in the following sections).

In non-preemptive scheduling (such as in IEEE 802.1Q), the worst-case transmission latency R_i^+ of a frame of stream i depends on the worst-case queueing delay at its corresponding output port. More precisely, the worst-case transmission latency of the q-th frame $R_i(q)$ can be derived from the worst-case queueing delay of the q-th frame in a busy period:

Definition 1. *At a given switch output port, the worst-case queueing delay $w_i(q,a_i^q)$ of the q-th frame of a traffic stream*

i, which arrived at time a_i^q, is the time interval from the instant the first frame of traffic stream i, which starts the level-i busy period [15] arrives at the output port's scheduler, until the q-th frame can be transmitted.

Note that, later in this section, we will explain why $w_i(q,a_i^q)$ depends on the arrival time a_i^q of the q-th frame. Also note that a_i^q is measured relative to the beginning of the busy period.

At an output port's scheduler, frames of a traffic stream i are exposed to interference by frames of other traffic streams. To calculate the worst-case queuing delay $w_i(q,a_i^q)$, we have to consider several blocking effects.

Lower-priority blocking: In non-preemptive scheduling, the transmission of a frame of stream i can be blocked by at most one lower-priority frame, if this lower-priority frame starts transmitting just before the first frame of stream i arrives.

$$I_i^{LPB} = \max_{j \in lp(i)} \{C_j^+\} \tag{2}$$

where $lp(i)$ is the set of all traffic streams with a priority lower than that of stream i.

Higher-priority blocking: In any time interval Δt, a frame of stream i can be blocked at most by all higher-priority frames, which arrive before this frame can be transmitted.

$$I_i^{HPB}(\Delta t) = \sum_{j \in hp(i)} \eta_j^+(\Delta t) C_j^+ \tag{3}$$

where $hp(i)$ is the set of all traffic streams with a priority higher than that of stream i.

Same-priority blocking: A frame of stream i is subject to blocking by frames of streams of equal priority. The q-th arrival of a frame of stream i, which arrived at time a_i^q, has to wait for its own $q-1$ predecessors to finish and for all frames of other same-priority streams, which have been queued previous to its arrival.

$$I_i^{SPB}(q,a_i^q) = (q-1)C_i^+ + \sum_{j \in sp(i)} \eta_j^{+]}(a_i^q) C_j^+ \tag{4}$$

Here $sp(i)$ is the set of all traffic streams with a priority equal to that of stream i (excluding stream i) and $\eta^{+]}(\Delta t)$ yields the number of frame arrivals in any *closed* time interval $[t, t+\Delta t]$, i.e. if frames arrive concurrently at exactly a_i^q, we assume the worst-case order.

As Ethernet typically serves frames of equal priority in FIFO order, a candidate search is required to compute the worst-case blocking [11]. The reason is that the earlier a frame arrives (within the bounds of its jitter), the longer its transmission latency might be, and the later a frame arrives (within the bounds of its jitter), the more blocking from previously queued same-priority frames it might experience. In [11] it has been shown that the set of arrival candidates a_i^q to consider for the q-th arrival of a frame of stream i can be reduced to the instances, where a_i^q coincides with arrivals of interfering same-priority frames.

$$A_i^q = \bigcup_{j \in sp(i)} \{\delta_j^-(n) | \delta_i^-(q) \le \delta_j^-(n) < \delta_i^-(q+1)\}_{n \ge 1} \tag{5}$$

Now, the worst-case queueing delay $w_i(q,a_i^q)$ for the q-th arrival of a frame of stream i, which arrived at time a_i^q, can be computed:

$$w_i(q,a_i^q) = I_i^{LPB} + I_i^{SPB}(q,a_i^q) + I_i^{HPB}(w_i(q,a_i^q)) \tag{6}$$

As $w_i(q,a_i^q)$ occurs on both sides, Eq. (6) cannot be solved directly. However, it represents a fixed-point problem, which can be solved by iteration, as all terms are monotonically increasing. A valid starting point is $w_i(q,a_i^q) = (q-1)C_i^+$.

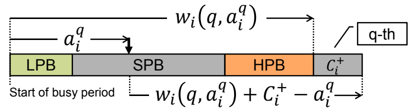


Fig. 1. Example queuing delay

From the worst-case queueing delay the largest transmission latency $R_i(q)$ for the q-th arrival of a frame of stream i can be derived by adding the execution time of the q-th frame C_i^+ and accounting for the fact that the q-th frame arrived at time a_i^q (see Figure 1).

$$R_i(q) = \max_{a_i^q \in A_i^q} \{w_i(q,a_i^q) + C_i^+ - a_i^q\} \tag{7}$$

The worst-case frame transmission latency for a frame of stream i is the maximum over all $R_i(q)$.

$$R_i^+ = \max_{1 \le q \le \hat{q}_i} \{R_i(q)\} \tag{8}$$

where $\hat{q}_i$ is the maximum number of frame arrivals of stream i which have to be evaluated. According to [15], for non-preemptive static-priority scheduling, $\hat{q}_i$ corresponds to the maximum number of frame transmissions of stream i in the longest level-i busy period. This is the longest time interval during which a port is processing frames with the priority of stream i (including interference from other streams). For IEEE 802.1Q, an upper bound on this longest level-i busy period can be computed similar to the queueing delay.

$$\hat{w}_i = I_i^{LPB} + \hat{I}_i^{SPB}(\hat{w}_i) + I_i^{HPB}(\hat{w}_i) \tag{9}$$

where, in contrast to Eq. (4), we do not need to distinguish between individual frames arrivals q. Hence, $\hat{I}_i^{SPB}(\Delta t)$ is defined as:

$$\hat{I}_i^{SPB}(\Delta t) = \sum_{j \in sp(i) \cup \{i\}} \eta_j^+(\Delta t) C_j^+ \tag{10}$$

Again, Eq. (9) represents a fixed-point problem, which can be solved by iteration, starting with an initial value of $\hat{w}_i = C_i^+$. Hence, the maximum number of frame arrivals of stream i, which have to be evaluated in Eq. (8), is $\hat{q}_i = \eta_i^+(\hat{w}_i)$.

V. Time-Aware Shaper (TSN/TAS)

In this section, we present a local analysis for TSN's time-aware shaper (TAS) [2]. TSN/TAS uses time-driven scheduling to manage link access between traffic classes. For each traffic class, the scheduler at an output port contains a gate, which allows frames to pass when opened and blocks frames when closed. The times at which these gates open and close are programmable (gate schedule). Gates of multiple traffic classes can be open concurrently. Then, link access is arbitrated according to the priority of these classes. To prevent frames of a traffic class from being transmitted after its gate closed, TSN/TAS defines *guard bands*. From the start of a guard band until the gate is closed, no new frames of the corresponding class are allowed to start transmission.

TSN/TAS suggests that each critical traffic class only has link access during special (scheduled) time intervals, i.e. the gate schedule is programmed such that only during these intervals the traffic class' gate is open and that it is closed otherwise [2]. Following [2], we also assume that these intervals are non-overlapping, so that every critical traffic class has exclusive link access during its intervals, i.e. without interference by higher- or lower-priority traffic. However, there still can be interference from same-priority traffic. This same-priority interference can be mitigated by further dividing these time intervals into (per frame) slots, e.g. like in TTEthernet

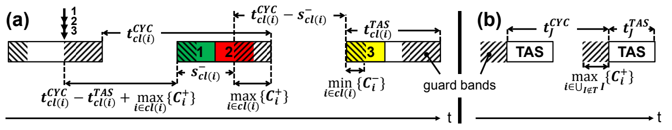

(a)
1
2
3
$t^{CYC}_{cl(i)}$
$t^{CYC}_{cl(i)} - s^{-}_{cl(i)}$
$t^{TAS}_{cl(i)}$
$s^{-}_{cl(i)}$
$t^{CYC}_{cl(i)} - t^{TAS}_{cl(i)} + \max_{i \in cl(i)}\{C_i^+\}$
$\max_{i \in cl(i)}\{C_i^+\}$
$\min_{i \in cl(i)}\{C_i^-\}$
guard bands
t
(b)
t^{CYC}_{J}
t^{TAS}_{J}
TAS
TAS
$\max_{i \in \cup_{I \notin T} I}\{C_i^+\}$
t

Fig. 2. TSN/TAS: (a) Common notations. Frame 3 experiences same-priority blocking from Frames 1 and 2. (b) Maximum blocking by a single TAS class.

or FlexRay's static segment. Such behavior, however, is not specified by the current version of the TSN/TAS standard. Outside these intervals, the gates of all other (non-critical) traffic classes are open, i.e. traffic from these other classes competes for link access according to its priority.

Let $\mathcal{T}$ be the set of all Ethernet traffic classes, which receive exclusive link access by the time-aware shaper. We will call traffic classes $I \in \mathcal{T}$ *TAS classes* and, correspondingly, their traffic streams *TAS streams* and frames of these streams *TAS frames*. Each TAS class I is associated with a time interval t_I^{TAS} (*TAS interval*), which is started every t_I^{CYC} time units (*TAS cycle*) (Figure 2a). In this interval, streams of TAS class I have exclusive access to the link of the output port. Traffic from all other streams (including other TAS classes) is gated early enough by a guard band before the start of this interval, such that the transmission of frames not belonging to streams of class I cannot overlap with I's TAS interval (Figure 2b). Note that TSN/TAS does not require TAS intervals to be periodic or of equal length (the gate schedule, however, will repeat eventually). We chose this periodic approach, because it naturally fits many automotive use cases. Our analysis approach can be extended to also support other gate schedules.

Next, we present two analyses for TSN/TAS: one to derive the worst-case transmission latency bounds for frames belonging to streams of a TAS class, and one to derive the worst-case transmission latency bounds for non-TAS frames.

A. Analysis of Time-Aware Traffic Streams

The frames of all TAS streams i of a TAS class I have exclusive access to an output port during class I's TAS interval and, while they are waiting for their (first or next) TAS interval to start, they are blocked. Inside their TAS interval, they only experience same-priority blocking from frames of streams $j \in cl(i)$, e.g. when frames belonging to class $cl(i)$ arrive concurrently at an output port from different input ports (e.g. same-priority frames 1 to 3 in Figure 2a). Where the function $cl(i)$ maps a traffic stream i to its corresponding traffic class. Thus, to compute the worst-case queueing delay $w_i(q, a_i^q)$ (see Definition 1) under TSN/TAS we have to consider same-priority and what we will call closed-gate blocking.

Same-priority blocking can be computed by Eq. (4), as it is independent of the number of TAS intervals required to serve q frames of stream i.

Closed-gate blocking: We model the blocking time between consecutive TAS intervals of a TAS class I as the time interval during which I's gate is closed plus the fraction of the TAS interval's guard band which might be left unused because of non-preemption. Then, the amount of closed-gate blocking can be derived from the number of TAS intervals required to serve the accumulated workload requested by TAS class I. Non-preemption turns this into a combinatorially complex problem: find the worst-case combination among the arriving TAS frames such that blocking (i.e. the number of TAS intervals) is maximized, while each interval (except for the last one) is utilized such that no other TAS frame can start transmission without finishing after the gate has closed. Note that this is not a bin packing problem, as bin packing would try to find the minimum number of TAS intervals (bins), i.e. the minimum blocking. We transform this discrete problem into a continuous one.

Theorem 1. *The minimum workload, which can be processed in a TAS interval of TAS class $cl(i) = I \in \mathcal{T}$ is given by:*

$$s^{-}_{cl(i)} = \max\Big\{ \underbrace{t^{TAS}_{cl(i)} - \max_{i \in cl(i)}\{C_i^+\}}_{(a)}, \underbrace{\min_{i \in cl(i)}\{C_i^-\}}_{(b)} \Big\} \qquad (11)$$

Proof. Term (a): To prevent the largest frame of TAS class $cl(i)$ from transmitting beyond its TAS interval boundaries, the guard band must be of size $\max_{i \in cl(i)}\{C_i^+\}$. As long as there is backlogged workload, each TAS interval is at least utilized for $t^{TAS}_{cl(i)} - \max_{i \in cl(i)}\{C_i^+\}$. After this time interval the largest frame would not fit into the TAS interval anymore. Hence, the minimum workload, which can be processed in a TAS interval can be lower bounded by term (a). Frames, whose transmission extends beyond the interval $t^{TAS}_{cl(i)} - \max_{i \in cl(i)}\{C_i^+\}$ into the guard band are still allowed (see frame 2 in Figure 2a), but their overlap into the guard band is not considered to be part of the minimum workload in term (a). This is a conservative overapproximation of the discrete problem. *Term (b)*: In any reasonable sized TAS interval, i.e. $t^{TAS}_{cl(i)} \geq \max_{i \in cl(i)}\{C_i^+\}$, at least one minimum-sized frame can be transmitted. As (a) and (b) are conservative bounds, we can take their maximum. □

Let the accumulated workload requested by TAS class $cl(i)$ be Δw. We can compute the maximum number of required TAS intervals by dividing Δw by $s^{-}_{cl(i)}$. The maximum time interval between two consecutive TAS intervals during which no frames of $cl(i)$ are transmitted is $t^{CYC}_{cl(i)} - s^{-}_{cl(i)}$ (see Figure 2a). The maximum closed-gate blocking for any frame of stream i is then given by:

$$I_i^{CGB}(\Delta w) = \left(\left\lceil \frac{\Delta w}{s^{-}_{cl(i)}} \right\rceil - 1 \right) \left(t^{CYC}_{cl(i)} - s^{-}_{cl(i)} \right) + t^{CYC}_{cl(i)} - t^{TAS}_{cl(i)} + \max_{i \in cl(i)}\{C_i^+\} \qquad (12)$$

where the last three terms are an upper bound on the largest time until the first (usable) TAS interval, i.e. in the worst-case we start with the largest frame, which arrived just after the guard band started (see Figure 2a). Note that this assumes that, in the worst-case, the first frame of a busy period just missed its TAS interval and has to wait for the next one. This worst-case scenario can occur, if the gate schedules throughout the network are not synchronized. Typically, one would engineer networks with TSN/TAS such that the gate schedules of all ports are synchronized and frames do not suffer from closed-gate blocking, i.e. TAS intervals are sufficiently large and aligned to each other. This, however, has synchronization implications on all switches and nodes in the network and also limits topology and traffic flow choices (e.g. rejoining paths becomes a complex task). We will additionally consider such a setup in our experiments.

With Eqs. (4) and (12), the worst-case queueing delay of the q-th TAS frame, which arrived at time a_i^q, becomes:

$$w_i(q, a_i^q) = I_i^{SPB}(q, a_i^q) + I_i^{CGB}\left(I_i^{SPB}(q, a_i^q) + C_i^+ \right) \qquad (13)$$

As $I_i^{CGB}(\Delta w)$ requires the accumulated workload requested by TAS class $cl(i)$ as its argument and $I_i^{SPB}(q, a_i^q)$ only considers $(q-1)C_i^+$, we must add one C_i^+. The maximum frame transmission latency can be computed similar to IEEE 802.1Q.

B. Analysis of Non Time-Aware Traffic Streams

Frames of non-TAS traffic streams $i \in \bigcup_{I \notin \mathcal{T}} I$ interfere with all other lower- and higher-priority streams of other non-TAS classes. They are not part of any TAS interval (non-TAS gates are open while all TAS gates are closed) and receive the left over bandwidth. Hence, we can model non-TAS traffic like IEEE 802.1Q and consider TAS intervals from all classes $J \in \mathcal{T}$ as additional periodic blocking terms, when deriving the worst-case queueing delay.

Specifically, **same-**, **lower-**, and **higher-priority blocking** from non-TAS traffic can be modeled as in Section IV, while bearing in mind that only interfering traffic from non-TAS traffic must be considered for lower- and higher-priority blocking.

Blocking by TAS classes: We start by bounding the maximum interference a single TAS interval can cause.

Theorem 2. *The maximum blocking caused by a single TAS interval t_J^{TAS} on a non-TAS class $I \notin \mathcal{T}$ is*

$$\tilde{t}_{I,J}^{TAS} = \max_{i \in \bigcup_{I \notin \mathcal{T}} I} \{C_i^+\} + t_J^{TAS} \tag{14}$$

Proof. The first term models that, to guard the TAS interval of TAS class J, the guard band of non-TAS class I must be large enough such that even the longest non-TAS frame cannot overlap into J's TAS interval (Figure 2b). The argumentation why this is conservative follows that of term (a) from Theorem 1. The second term is the TAS interval of J. □

In any time interval Δt, the maximum number of times a TAS class J can interfere with frames of a non-TAS stream i can be derived by diving Δt by t_J^{CYC}. With Eq. (14) the interference from all TAS classes $J \in \mathcal{T}$ on a non-TAS stream i becomes:

$$I_i^{TASB}(\Delta t) = \sum_{J \in \mathcal{T}} \left\lceil \frac{\Delta t}{t_J^{CYC}} \right\rceil \tilde{t}_{cl(i),J}^{TAS} \tag{15}$$

where we conservatively assume that, in the worst-case, the blocking by different TAS classes does not overlap.

With Eqs. (4), (2), (3), and (15) the worst-case queueing delay of the q-th non-TAS frame, which arrived at time a_i^q, becomes:

$$w_i(q,a_i^q) = I_i^{LPB} + I_i^{SPB}(q,a_i^q) + I_i^{HPB}(w_i(q,a_i^q)) + I_i^{TASB}(w_i(q,a_i^q)) \tag{16}$$

Again, the maximum frame transmission latency can be computed similar to that of IEEE 802.1Q in Section IV.

VI. Peristaltic Shaper (TSN/PS)

This section presents a local analysis for TSN's peristaltic shaper (PS) [16]. In addition to their priority, TSN/PS classifies frames based on their arrival time. It divides time in alternating (peristaltic) intervals of equal size (*even*, *odd*). Frames received in an even interval are scheduled to be transmitted in the next odd interval and vice versa, based only on their arrival interval. If more frames need to be transmitted than time is available in the next interval (transient overload), frame transmission from the overloaded interval continues and overlaps into subsequent intervals (Figure 3). In this case the transmission of frames that are scheduled to be transmitted in these subsequent intervals is delayed until the overload has been processed. This way the intra-class frame order is preserved (even) under transient overload. The peristaltic interval pattern, however, is left untouched. The motivation for

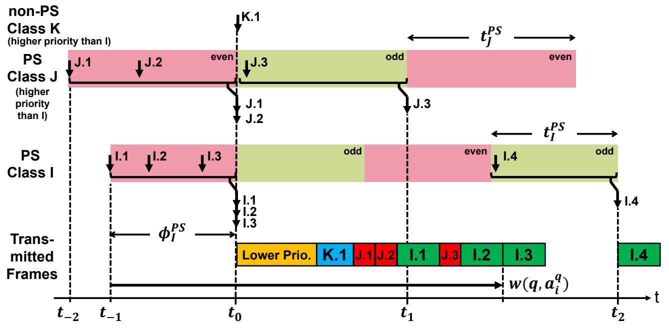


Fig. 3. TSN/PS: Frames of PS class I experience interference from higher priority classes K (non-PS) and J (PS) and one lower-priority frame. It takes more than one PS interval to transmit the first three frames of PS stream I.

TSN/PS is to bound the residence time of frames in the switch for easier latency computation. However, this only works in the absence of any interfering traffic. In reality, a thorough timing analysis is still required.

Let $\mathcal{P}$ be the set of peristaltic (PS) traffic classes, i.e. the set of all Ethernet classes which are shaped by a peristaltic shaper based on their arrival interval. All streams of other (non-PS) classes are treated as in IEEE 802.1Q. We call traffic classes $I \in \mathcal{P}$ *PS classes* and, correspondingly, their traffic streams *PS streams* and frames of these streams *PS frames*. Further, let t_I^{PS} be the *peristaltic (PS) interval* length of a given PS class $I \in \mathcal{P}$. Note that, like Ethernet AVB's CBS and TSN/TAS, TSN/PS is not work-conserving. In TSN/PS, after a frame of PS stream i arrives at an output port, it must wait for its current PS interval to end until it can take part in the arbitration process of the port's scheduler, regardless of whether the port is busy processing other frames during this waiting time or not. The waiting time of first frame that starts a busy period defines the PS interval pattern (relative to its arrival) for the rest of the busy period. We will call the waiting time of such a first frame $i \in I$ the initial *PS offset* ϕ_I^{PS} of PS class I (Figure 3).

Next, we present two analyses for TSN/PS: one to derive the worst-case transmission latency bounds for frames belonging to streams of a non-PS class, and one to derive the worst-case transmission latency bounds for PS frames.

A. Analysis of Non Peristaltic Traffic Streams

Non-PS traffic is scheduled as in IEEE 802.1Q. Hence, **same-priority blocking** can be computed as in IEEE 802.1Q, when deriving the worst-case queueing delay. **Lower-priority blocking** only considers the largest lower-priority blocker, independently of whether this blocker comes from a PS class or not. Therefore, it can also be computed as in IEEE 802.1Q.

Higher-priority blocking, however, requires special attention, as there are two different cases: (a) the higher-priority blocking is by frames of non-PS streams j, i.e. $j \in hp(i) \wedge cl(j) \notin \mathcal{P}$, and (b) the higher-priority blocking is by frames of PS streams j, i.e. $j \in hp(i) \wedge cl(j) \in \mathcal{P}$.

In case (a), higher-priority blocking on frames of stream i can be computed as in IEEE 802.1Q.

$$I_i^{HPB,nPS}(\Delta t) = \sum_{j \in \{k | k \in hp(i) \wedge cl(k) \notin \mathcal{P}\}} \eta_j^+(\Delta t) C_j^+ \tag{17}$$

In case (b), we have to consider that frames of the streams of a PS class $J \in \mathcal{P}$ might not be presented to the scheduler immediately. In the worst-case, these frames have been delayed for the length of their PS interval t_J^{PS}, before they can interfere with the frames of stream i. Hence, interference from PS streams occurs at the end of their PS intervals. This is shown in Figure 3 for PS class J, where, for example, frames $J.1$ and $J.2$ are delayed during a PS interval until they are released concurrently at the interval's end. For any time interval Δt, the maximum number of PS intervals of PS class J, which end in this Δt, can be computed by $\lceil \Delta t / t_J^{PS} \rceil$. Multiplying this number by the PS interval length of PS class J yields the time interval during which the interference from PS streams $j \in J$ has been accumulated, i.e. $\lceil \Delta t / t_J^{PS} \rceil t_J^{PS}$. We can use this time interval to compute the higher-priority blocking of each stream j of PS class J on stream i.

$$I_i^{HPB,PS}(\Delta t) = \sum_{j \in \{k | k \in hp(i) \wedge cl(k) \in \mathcal{P}\}} \eta_j^+ \left(\left\lceil \frac{\Delta t}{t_{cl(j)}^{PS}} \right\rceil t_{cl(j)}^{PS} \right) C_j^+ \tag{18}$$

The worst-case queueing delay and the maximum frame transmission latency can be computed as for IEEE 802.1Q, with the sum of Eqs. (17) and (18) replacing Eq. (3).

B. Analysis of Peristaltic Traffic Streams

The analysis of a PS traffic stream i is similar to the analysis of non-PS traffic, except for the initial offset $\phi_{cl(i)}^{PS}$. This initial offset only affects streams of the currently analyzed PS class $cl(i)$. All streams of other (interfering) traffic classes can potentially transmit frames during this time. Hence, the interference on PS stream i is maximized if interfering traffic arrives at the end of i's initial offset, i.e. directly after $\phi_{cl(i)}^{PS}$ relative to the beginning of stream i's busy period (e.g. at t_0 in Figure 3). Any frames of streams of potentially interfering traffic classes arriving before this point would be transmitted without interfering with PS stream i (only after $\phi_{cl(i)}^{PS}$ their transmission would affect PS stream i). Concretely, in the worst-case, non-PS interferers start interfering with PS stream i directly after $\phi_{cl(i)}^{PS}$ and the end of the first PS intervals of PS interferers is aligned to the end of $\phi_{cl(i)}^{PS}$ (e.g. t_0 in Figure 3).

Lower-priority blocking can be computed as in the non-PS case. For PS streams i, it starts interfering with i just after $\phi_{cl(i)}^{PS}$, but the amount of interference is independent of $\phi_{cl(i)}^{PS}$.

Same-priority blocking is also independent of $\phi_{cl(i)}^{PS}$, since the amount of same-priority blocking only depends on the $q-1$ preceding frames of stream i itself and all frames of class $cl(i)$ that have arrived until arrival candidate a_i^q (see Eq. (4)).

For **higher-priority blocking**, the argumentation from the non-PS analysis also holds for the PS analysis and Eqs. (17) and (18) can be used. However, as argued above, any frames of interfering traffic streams that are released before $\phi_{cl(i)}^{PS}$, do not interfere with PS stream i during $\phi_{cl(i)}^{PS}$. Hence, higher-priority blocking is maximized if we align interfering higher-priority traffic to $\phi_{cl(i)}^{PS}$, so that the actual higher-priority interference starts directly after $\phi_{cl(i)}^{PS}$ relative to the beginning of the busy period. In Figure 3, for example, PS class J is aligned such that frames $J.1$ and $J.2$ from the first PS interval of J start interfering with PS class I at t_0, i.e. just when ϕ_I^{PS} ended.

Finally, the worst-case queueing delay can be computed by taking into account the length of the initial offset $\phi_{cl(i)}^{PS}$ as an additional blocking term of PS stream i.

$$w_i(q,a_i^q) = \phi_{cl(i)}^{PS} + I_i^{LPB} + I_i^{HPB,nPS}\left(w_i(q,a_i^q) - \phi_{cl(i)}^{PS}\right) + I_i^{HPB,PS}\left(w_i(q,a_i^q) - \phi_{cl(i)}^{PS}\right) + I_i^{SPB}(q,a_i^q) \tag{19}$$

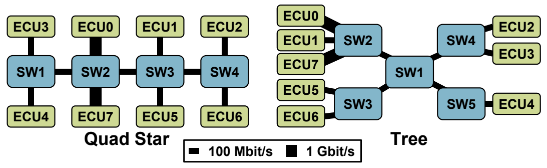


Fig. 4. Quad star and tree topologies

Note that $\phi_{cl(i)}^{PS}$ is an additive term to model the initial waiting time that is experienced only by the analyzed PS stream i and streams of $cl(i)$. As interference starts only after $\phi_{cl(i)}^{PS}$ (relative to the beginning of $w_i(q,a_i^q)$), interference should not be considered during $\phi_{cl(i)}^{PS}$. Hence, we must subtract $\phi_{cl(i)}^{PS}$ from the time intervals of both higher-priority blocking terms. The amount of lower- and same-priority blocking is independent of $\phi_{cl(i)}^{PS}$ (see above). In Figure 3, for example, ϕ_I^{PS} is subtracted from $w_i(q,a_i^q)$ so that the time interval, in which higher-priority blocking from PS class J according to Eq. (18) must be considered, starts directly after ϕ_I^{PS} at t_0.

Now, we show which offset maximizes the queueing delay.

Theorem 3. *A PS traffic stream i experiences its longest queueing delay $w_i(q,a_i^q)$, if $\phi_{cl(i)}^{PS} = t_{cl(i)}^{PS}$.*

Proof. As the PS interval pattern repeats periodically, any offset by more than $t_{cl(i)}^{PS}$ and any negative offset is already covered by an offset $\phi_{cl(i)}^{PS} \in (0, t_{cl(i)}^{PS}]$. It can be seen, that the queueing delay in Eq. (19) is largest, when $\phi_{cl(i)}^{PS} = t_{cl(i)}^{PS}$. □

In TSN/PS, frames of streams of a PS class I that arrive during a certain PS interval of I are not released before the end of this PS interval. Consequently, frames of a PS traffic stream under analysis that arrive in the last PS interval of its busy period do not contribute to this busy period anymore, as the corresponding output port is idle before these frames are considered by its scheduler. Instead, these frames start a new busy period. This is also the case when the end of the busy period coincides with the end of the PS interval, i.e. the corresponding output port's scheduler is idle for an infinitesimal small time interval. In Figure 3, for example, frame $I.4$, even though it arrives within the busy period started by the arrival of frame $I.1$, is not part of this busy period, as the port is idle before $I.4$ can be scheduled (i.e. before t_2). Our TSN/PS analysis takes this into account when it computes the arrival candidate set A_i^q in Eq. (5) and the maximum number $\hat{q}_i$ of frame arrivals of PS stream i to consider in Eq. (8).

Note that, according to Eqs. (18) and (19) (with $\phi_{cl(i)}^{PS} = t_{cl(i)}^{PS}$), if all t_I^{PS} approach 0, TSN/PS becomes IEEE 802.1Q.

VII. Experiments

In this section, we evaluate the worst-case end-to-end latency guarantees of TSN/TAS and TSN/PS and compare them to standard Ethernet with priorities (IEEE 802.1Q) by using our proposed analysis method. As suggested by [12], we extended our analyses to take into account that an Ethernet link can be seen as an additional shaper and limits the amount of workload (bits) which can pass it in a given amount of time.

Our evaluation focuses on the topologies in Figure 4. A fixed number of eight ECUs is distributed throughout the network. The number of switches is topology-dependent. Highly loaded links operate at 1Gbit/s. All other links are 100Mbit/s links. The network traffic is summarized in Table I, which has been provided by Daimler AG. The traffic in

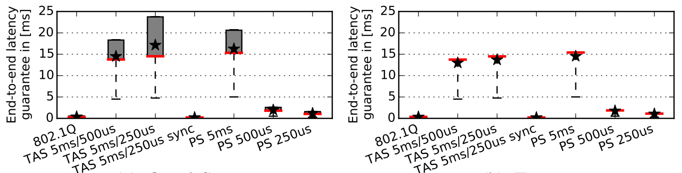
Table I is grouped into three classes: control data traffic (CDT), general control traffic (GCT), and camera traffic (CAM). Traffic streams belonging to the CDT class are latency-critical and, hence, CDT traffic is mapped to the highest Ethernet priority and uses TSN's shaping mechanisms. GCT traffic streams, while still latency sensitive, are not as critical as CDT traffic and are mapped to a priority below CDT. CAM traffic consists of high-bandwidth video traffic with comparatively relaxed latency requirements and is mapped to a priority below GCT. GCT and CAM do not use TSN's shaping mechanisms and are scheduled only according to their priority. There is unicast, multicast, and broadcast traffic in the network. Multicast traffic uses the notation $n(d)$, which indicates that there are n multicast streams with d targets. In our evaluation we consider each path in the multicast and broadcast trees as an individual path. Hence, the worst-case latency guarantees of a total of 115 paths are compared. Payloads and periods are characterized by their average, minimum, and maximum values. Note that all CDT traffic streams have a period of 5ms. The payload is given as the raw application payload. We assume that IPv4/UDP is used to route the traffic, such that an additional amount of 28bytes for the protocol overhead must be added. During our experiments, we assume that ECUs send their traffic streams according to a *periodic with jitter* event model [4]. Even though we assume periodic traffic streams, we set each stream's jitter to its period in order to also model that, in the worst-case, there might be a burst of two frames entering the network. Additionally, each Ethernet link is assumed to contribute 33ns of transmission delay to the end-to-end latency, which corresponds to wires of 10m length. Next, we evaluate the impact of each TSN shaping mechanism on CDT traffic. Then, we investigate the impact of the shaped CDT traffic on unshaped lower-priority GCT and CAM traffic.

Figure 5 shows the worst-case end-to-end latency guarantees from our analysis for different TSN shapers against the guarantees of IEEE 802.1Q. Although this is not a random experiment, we use boxplots to efficiently summarize the latency guarantees of the 115 analyzed paths. The box covers 50% of the path latency guarantees with its lower and upper edges representing the 25% and 75% quartiles. The whiskers indicate the minimum and maximum worst-case latency guarantees among all paths. The median is marked by a red bar and the average by a star. Figure 6 provides a more detailed view on the 4 shaper configurations with the lowest latency guarantees.

We briefly discuss the difference in end-to-end latency guarantees between the quad star and the tree topologies. The tree topology shows lower latency guarantees for CDT traffic (Figure 5). This is because, compared to the quad star, there is less lower-priority interference from CAM traffic and (for example) the length of a path (ECU2→ECU4) carrying 5 CDT streams is shortened from 4 to 3 switches. The latency guarantees for GCT and CAM traffic, in contrast, are worse than in the quad star (Figure 7). This is mainly because the path lengths of two CAM streams (both ECU7→ECU4) are extended from 2 to 3 switches. The main focus of this paper, however, is on the analytical worst-case performance of the TSN shapers. Comparing both topologies, we see that, for each topology, the relative (average) performance of the TSN shaper configurations among each other almost stays the same, e.g. *PS 250us* gives lower average latency guarantees than *PS 500us* regardless of the topology.

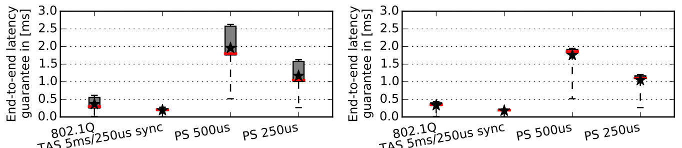
TABLE I
TRAFFIC CHARACTERISTICS

	CDT	GCT	CAM
Unicast (#)	8	18	3
Multicast (#)	2(2)	11(2), 4(3), 1(4)	1(2)
Broadcast (#)	0	6	0
Payload (bytes)	[14, 144]	[1, 250]	[875, 1400]
Average (bytes)	70	50	1231
Period	5ms	[10ms, 1s]	[100us, 1ms]
Average	5ms	230ms	440us



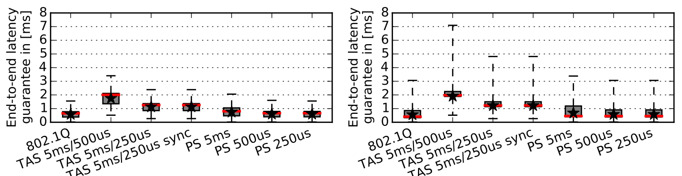
(a) Quad Star (b) Tree

Fig. 5. Worst-case end-to-end latency guarantees for CDT traffic



(a) Quad Star (b) Tree

Fig. 6. Worst-case end-to-end latency guarantees for CDT traffic focusing on 802.1Q, TAS 5ms/250us sync, PS 500us, and PS 250us



(a) Quad Star (b) Tree

Fig. 7. Worst-case end-to-end latency guarantees for GCT and CAM traffic

For TSN/TAS we evaluate three shaper configurations, which are applied to all switch ports: *TAS 5ms/500us*, *TAS 5ms/250us*, and *TAS 5ms/250us sync*. The first parameter of each configuration is t^{CYC} and is set to the CDT period of 5ms. The second parameter is t^{TAS}, which is evaluated for 500us and 250us. The last configuration assumes synchronized shapers (gate schedules) throughout the network, while the first two are unsynchronized. Our analysis approach requires the TSN/TAS guard bands for TAS and non-TAS classes to be set to their maximum frame length, respectively (see Eqs. (11) and (14)). The overhead introduced by this on the evaluated configurations is small: the guard bands for TAS traffic account for 3.4% (for t^{TAS}=500us) and 6.8% (for t^{TAS}=250us) of the TAS interval length. The guard bands for non-TAS traffic account for 2.6% and 2.5% (respectively) of the interval length, in which non-TAS traffic is processed.

First, we discuss the unsynchronized configurations with TAS interval lengths of 500us and 250us. Under IEEE 802.1Q, on each switch, a time interval of 250us is enough to process the longest busy period of CDT traffic. In TSN/TAS, however, this time is not sufficient. TSN/TAS suffers from long blocking times (4.5ms and 4.8ms in our setups, if a TAS frame just missed its TAS interval (see Eq. (12))). This leads to large jitter values, which cause large transient loads on subsequent switch ports, which, in turn, require a longer time to be processed. In the worst-case this load cannot be processed in a single TAS interval, which leads to even larger jitter. This is the reason why the *TAS 5ms/500us* configuration (t^{TAS}=500us), which offers more processing time per TAS interval, performs better than *TAS 5ms/250us* (t^{TAS}=250us). To avoid this jitter build up effect, the TAS intervals along a communication path should grow in size in unsynchronized TSN/TAS setups.

The third configuration does not suffer from long blocking times, as it assumes that all TAS frames arrive just in time for their TAS interval and, hence, only experience same-priority blocking. This results in significantly better latency guarantees. In this case, our analysis checks that t^{TAS} is long enough to process all incoming frames within a single TAS interval. Bare in mind that, in this paper, we focus on the end-to-end latency guarantees in the network. A system-wide end-to-end analysis, however, might involve additional components, e.g. the (software) communication stack in the ECUs. Also, there might be non-periodic latency-critical communication, e.g. spontaneous (emergency breaking) or angular-synchronized (engine control). Hence, even if the Ethernet communication is synchronized, data might be delayed (and experience jitter) in other components, such that it might, in fact, not be (perfectly) synchronized to its designated TAS interval. Synchronization to the TAS intervals introduces additional delay (sampling delay), when data is handed over (from ECUs) to the network.

Next we discuss TSN/PS by considering three configurations, which are applied to all switch ports: *PS 5ms*, *PS 500us*, and *PS 250us*. The parameter of these configurations corresponds to the PS interval length t^{PS} of the PS traffic class. As TSN/PS, in the worst-case, just artificially delays traffic by its PS interval length (and otherwise behaves like IEEE 802.1Q), shorter PS intervals lead to shorter delays and smaller jitter, which, in turn, result in lower worst-case latency guarantees. This is why *PS 250us* performs best among the evaluated TSN/PS configurations. It has been suggested to configure the PS interval length such that the longest busy period can be processed entirely within t^{PS} [16]. Depending on the traffic, this could require quite long PS intervals. As the worst-case jitter grows at least by the PS interval length per switch port, this can lead to large (transient) bursts, which, in turn, require (even) longer PS intervals on subsequent switches.

Figure 7 shows the end-to-end latency guarantees for (non-TSN) GCT and CAM traffic in the presence of (TSN) CDT traffic. In the evaluated setup, higher-priority traffic in IEEE 802.1Q and all TSN/PS configurations has almost the same impact on lower-priority traffic. This is because in TSN/PS, higher-priority blocking is modeled similar to higher-priority blocking in IEEE 802.1Q. The reason why the impact of TSN/TAS is higher is twofold: (a) Lower-priority (non-TAS) traffic experiences blocking by the TAS intervals of TAS traffic regardless of whether these TAS intervals are fully utilized or not (TAS intervals might even be designed with an additional safety margin to allow for occasional overload). (b) Guard bands reduce the available bandwidth. In our setup the guard bands of non-TAS traffic only account for about 2.5% of the intervals in which non-TAS traffic is processed. Hence, the blocking by the TAS intervals of TAS traffic is the dominating factor. As expected, this is independent of whether network-wide gate schedule synchronization is assumed or not.

Comparing Figures 5 and 6 to Figure 7 yields an interesting result. For unsynchronized TSN/TAS, the latency guarantees for almost all *unshaped* lower-priority GCT and CAM traffic are actually lower than the ones given for CDT traffic, as, in the worst-case, they have to wait less to receive service. This also holds for the *PS 5ms* configuration of TSN/PS. For TSN/PS with *PS 500us* most and for TSN/PS with *PS 250us* a few GCT and CAM traffic streams have lower latency guarantees than CDT traffic. In these cases it would, hence, make more sense to use non-TAS traffic classes or IEEE 802.1Q for latency-critical traffic rather than TSN's shapers.

VIII. Conclusion

In this paper, we presented a formal analysis approach to derive worst-case timing guarantees for the time-aware (TAS) and peristaltic shapers (PS) of the upcoming Ethernet TSN standard. In contrast to related work, our analysis considers all blocking effects of these shapers, which enables a thorough timing analysis for all traffic streams in Ethernet TSN setups. We used typical automotive network setups to evaluate our analysis and compared the results against standard Ethernet (IEEE 802.1Q). We evaluated the worst-case latency guarantees by TSN's shapers for latency-critical traffic, as well as their impact on unshaped lower-priority traffic. Our experiments show that, TSN/TAS can give very low latency guarantees when all shapers are synchronized. However, it suffers from long blocking times if this cannot be guaranteed, which underlines the need for synchronization in TSN/TAS setups. TSN/PS suffers from the (artificial) initial delay caused by its peristaltic interval. We showed, that TSN/PS's worst-case performance is always worse than that of IEEE 802.1Q.

References

[1] IEEE Audio Video Bridging Task Group, "802.1Qav - Forwarding and Queuing Enhancements for Time-Sensitive Streams," http://www.ieee802.org/1/pages/802.1av.html.
[2] IEEE Time-Sensitive Networking Task Group, "P802.1Qbv (Draft 3.0) - Enhancements for Scheduled Traffic," http://www.ieee802.org/1/pages/802.1bv.html.
[3] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello, "Simulative Assessments of IEEE 802.1 Ethernet AVB and Time-Triggered Ethernet for Advanced Driver Assistance Systems and In-car Infotainment," in *IEEE Vehicular Networking Conference (VNC)*, Seoul, South Korea, November 2012.
[4] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System Level Performance Analysis - the SymTA/S Approach," in *IEE Proceedings Computers and Digital Techniques*, 2005.
[5] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *The 27th Annual International Symposium on Computer Architecture (ISCA)*, vol. 4, 2000.
[6] J. Diemer, J. Rox, R. Ernst, F. Chen, K.-T. Kremer, and K. Richter, "Exploring the Worst-Case Timing of Ethernet AVB for Industrial Applications," in *Proc. of the 38th Annual Conference of the IEEE Industrial Electronics Society*, Montreal, Canada, October 2012.
[7] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of Ethernet-switch Traffic Shapers for In-vehicle Networking Applications," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015.
[8] J. Rox and R. Ernst, "Formal Timing Analysis of Full Duplex Switched Based Ethernet Network Architectures," in *SAE World Congress*, vol. System Level Architecture Design Tools and Methods (AE318), 2010.
[9] D. Thiele, P. Axer, and R. Ernst, "Improving Formal Timing Analysis of Switched Ethernet by Exploiting FIFO Scheduling," in *Design Automation Conference (DAC)*, San Francisco, CA, USA, June 2015.
[10] Avionic Systems Group AS-2D2, "TTEthernet (SAE AS6802)," http://standards.sae.org/as6802/.
[11] J. Diemer, D. Thiele, and R. Ernst, "Formal Worst-Case Timing Analysis of Ethernet Topologies with Strict-Priority and AVB Switching," in *IEEE International Symposium on Industrial Embedded Systems*, 2012.
[12] P. Axer, D. Thiele, R. Ernst, and J. Diemer, "Exploiting Shaper Context to Improve Performance Bounds of Ethernet AVB Networks," in *Proceedings of DAC*, San Francisco, 2014.
[13] K. W. Tindell, A. Burns, and A. J. Wellings, "An extensible approach for analysing fixed priority hard real-time tasks," *Real-Time Systems Journal*, vol. 6, pp. 133–151, 1994.
[14] J. Diemer, J. Rox, and R. Ernst, "Modeling of Ethernet AVB Networks for Worst-Case Timing Analysis," in *MATHMOD - Vienna International Conference on Mathematical Modelling*, Vienna, Austria, 2012.
[15] R. I. Davis and A. Burns, "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised," *Real-Time Systems*, vol. 35, 2007.
[16] M. J. Teener, "Back to the Future: Using TAS and Preemption for Deterministic Distributed Delays," in *IEEE 802.1 AVB TG Meeting*, San Antonio, TX, USA, November 2012.