Formal Worst-Case Timing Analysis of Ethernet TSN's Burst-Limiting Shaper

Daniel Thiele and Rolf Ernst
Institute of Computer and Network Engineering
Technische Universität Braunschweig, Germany
{thiele,ernst}@ida.ing.tu-bs.de

Abstract—**Future in-vehicle networks will use Ethernet as their communication backbone. As many automotive applications are latency-sensitive and have strict real-time requirements, a key challenge in automotive network design is the deterministic low-latency transport of latency-critical Ethernet frames. Time-sensitive networking (TSN) is an upcoming set of Ethernet standards, which address these requirements by specifying new quality of service mechanisms in the form of different traffic shapers. One of these traffic shapers is the burst-limiting shaper (BLS). In this paper, we evaluate whether BLS is able to fulfill these strict timing requirements. We present a formal timing analysis for BLS in order to compute worst-case latency bounds. We use a realistic automotive Ethernet setup to compare BLS against Ethernet AVB and Ethernet following IEEE 802.1Q.**

I. Introduction

The communication backbone of future automotive networks will be based on packet-switched Ethernet. Ethernet provides more bandwidth than traditional buses such as CAN or FlexRay, which struggle to keep pace with the increasing bandwidth demand of advanced driver assistance and infotainment systems. It also allows the implementation of a scalable and cost-effective communication infrastructure. As Ethernet has complex timing behavior (all switch output ports are points of arbitration), a thorough timing verification is required before it can be used in time- and safety-critical systems.

In order to provide quality of service (QoS) for networks with mixed-critical traffic, standard Ethernet (IEEE 802.1Q) introduced eight traffic classes, which can be used to prioritize traffic. Since there are typically more traffic streams than traffic classes, multiple traffic streams have to share a class. The scheduling within a class is usually FIFO. There are different Ethernet standards, each with its own QoS mechanism, which defines how frames from these (per-class) FIFO queues are arbitrated during link access. IEEE 802.1Q typically arbitrates between these FIFO queues based on the priority assigned to each queue (i.e. by static-priority non-preemptive scheduling). Ethernet AVB [1] introduced standardized traffic shaping for the two queues with the highest priorities by using a credit-based shaper (CBS), which enforces that shaped traffic classes do not exceed their preconfigured bandwidth limits. CBS-based traffic shaping, however, can lead to additional shaper delays on these two highest priorities (i.e. typically the most critical ones), which is undesirable for latency-critical traffic.

The upcoming set of Ethernet time-sensitive networking (TSN) standards [2] proposes new QoS mechanisms to provide deterministic end-to-end latencies for (hard) real-time traffic. Currently, TSN proposes three shapers: the time-aware shaper (TAS), the peristaltic shaper (PS), and the burst-limiting shaper (BLS). In TSN/TAS [2] critical traffic streams are scheduled in time-triggered segments similar to TTEthernet [3]. Very low network latencies can be achieved if these segments are synchronized properly throughout the entire network [4]. However, as in any time-triggered system, there might be an additional sampling delay (i.e. the waiting time until the next time-triggered segment) when unsynchronized data is passed from an end-point to the network. TSN/PS [5] tries to bound the residence time of frames in the switches by introducing peristaltic frame forwarding. It divides time into alternating (peristaltic) intervals (even and odd) of equal size. Frames received in an even interval are scheduled to be transmitted in the next odd one and vice-versa. In their designated interval, these frames are scheduled as in IEEE 802.1Q. It can be shown that the worst-case timing behavior of TSN/PS is very similar to IEEE 802.1Q with an additional delay of one peristaltic interval [4]. TSN/BLS [6] is similar to Ethernet AVB in that it also uses a credit-based shaper. However, AVB only allows bursts of critical traffic after blocking by lower-priority traffic, i.e. to help critical traffic to compensate for the lost blocking time. TSN/BLS, in contrast, allows bursts of limited size without prior blocking. The intention behind this is to allow TSN/BLS to react faster to transient overloads of critical traffic. In the worst-case, however, frames in TSN/BLS (just like frames of shaped AVB classes) must wait for their credit to replenish.

Traditionally, simulation has been used to evaluate network performance. Simulation, however, often requires long simulation runs and usually does not guarantee to expose all corner cases, rendering it insufficient for the timing verification of timing- and safety-critical systems. Formal performance analyses like compositional performance analysis (CPA) [7] or real-time calculus (RTC) [8], in contrast, have been proven to efficiently provide safe latency bounds for Ethernet networks. Formal analysis methods are becoming even more important with the advent of highly-automated and autonomous driving.

The **contribution of this paper** is a formal timing analysis for Ethernet TSN's burst-limiting traffic shaper. In contrast to related work [9], we consider all blocking effects, especially those of same-priority traffic streams. This is important, as Ethernet only offers a limited number of priorities. We evaluate the worst-case timing behavior of TSN/BLS and compare its latency guarantees against those of Ethernet AVB and IEEE 802.1Q by using a realistic automotive Ethernet setup.

II. Related Work

Over the last years, many formal timing analysis methods for Ethernet have been developed. Standard Ethernet, i.e. IEEE 802.1Q, has been analyzed by e.g. [10]. Formal analyses for Ethernet AVB [1] are presented in e.g. [11], [12], [13]. Formal analyses for TSN/TAS and TSN/PS are presented in [4].

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

A first attempt to formally analyze the timing behavior of TSN/BLS (and also TSN/TAS and TSN/PS) is proposed in [9]. This analysis, however, has two main limitations: (a) It does not consider interference by same-priority traffic. Same-priority interference occurs whenever frames of equal priority compete for link access and is very common in Ethernet, as Ethernet only offers a limited number of (up to) eight priorities. (b) It does not consider higher-priority interference on BLS traffic, i.e. it is only possible to analyze a single BLS traffic class, which must be on the highest priority. The TSN task group, however, envisions more than a single BLS class [14].

In contrast, we will present a formal analysis based on the proven CPA framework, which does not suffer from these limitations and, consequently, allows a thorough formal timing analysis of TSN/BLS.

III. Compositional Performance Analysis of Ethernet

In this paper, we use CPA to compute worst-case end-to-end latency guarantees for Ethernet frames [7], [15]. The CPA system model relies on three components: resources, tasks, and event models. *Resources* abstract the actual processing resources and provide service according to a scheduling policy. In Ethernet, we assume that the output ports of switches and end-points are the points of arbitration and, hence, we model them with resources [15]. *Tasks* are mapped to these resources and consume a certain amount of service. In Ethernet, we model the transmission of a frame at an output port as a task executing on the port's resource. The transmission times of Ethernet frames correspond to task execution times. They are defined to be the frame transmission times on a switch port in the absence of any interference. We assume that each Ethernet frame of a traffic stream i is of a predefined, fixed size C_i^+:

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

where p_i is the frame's payload size and r_{TX} is the port's transmission rate. The constant terms account for minimum Ethernet frame size, protocol overhead, VLAN tagging, and inter-frame gap. An Ethernet network is modeled as a directed graph. Tasks correspond to nodes and task dependencies correspond to edges. Whenever a task finishes executing, i.e. whenever a port transmits a frame, it sends an event to its dependent tasks. A traffic stream through an Ethernet network is modeled as a chain of tasks, where the edges between tasks represent the stream's path. Tasks without predecessors must be stimulated by events from external sources. *Event models* are used to abstract frame arrivals at tasks. They are defined as a tuple of event arrival functions $\eta_i^-(\Delta t)$ and $\eta_i^+(\Delta t)$, which describe the lower and upper bound on the number of frame arrivals in any half-open time interval $[t, t + \Delta t)$. Hence, in contrast to a single event trace, event models capture all possible frame arrival scenarios within their bounds. Alternatively, event models can be defined by the pseudo-inverse of their arrival functions, $\delta_i^+(n)$ and $\delta_i^-(n)$, which describe the maximum and minimum time between the first and the last event in any sequence of n events.

CPA is based on an iterative approach. A *local analysis*, which relies on the busy period approach [16], is used to analyze each resource to derive new output event models for each frame transmission. These event models are derived by constructing a critical instant scenario, which maximizes the transmission latency of the frame under analysis by spanning the busy period using worst-case arrival sequences of all interfering frames. The transmission latency jitter (maximum minus minimum transmission latency) on each resource can be used to compute new output event models [7]. The output event model of a frame transmission becomes the input event model of its dependent frame transmissions (i.e. tasks on subsequent ports). A *global analysis loop* is used to propagate event models. The analysis finishes, if all event models become stable. Otherwise the system is deemed unschedulable, e.g. if a predefined constraint (deadline, jitter, latency, etc.) is violated. CPA can also be used to compute jitter and buffer size bounds.

IV. Compositional Performance Analysis of Ethernet TSN's Burst-Limiting Shaper (TSN/BLS)

TSN/BLS limits the workload, which can be released by a traffic-shaped class in a given time interval, by a credit-based traffic shaper [6]. Let $\mathcal{B}$ be the set of all Ethernet traffic classes, which pass a burst-limiting shaper. We will call these classes *BLS classes* and, correspondingly, their traffic streams *BLS streams* and frames of these streams *BLS frames*. Each shaper of a BLS class I has a credit counter. Depending on the state of this shaper, I is either enabled or blocked[1]. If I is enabled, the frames in its FIFO queue take part in the link arbitration process according to their priority. Each time a frame of size C^+ is sent from I's FIFO queue, I's credit is incremented by a *send slope* $s_I^S > 0$, i.e. by $s_I^S C^+$. If class I is interrupted by interfering frames, its credit is decremented at a rate defined by an *idle slope* $s_I^I < 0$. An enabled class is allowed to send frames until its credit reaches a certain limit H_I. After H_I is reached, class I is blocked and credit replenishment is required. A frame of class I, which started transmission (just) before H_I was reached and whose credit increment would exceed H_I is allowed to finish, but I's credit is kept at H_I. Note that, while I is blocked, other traffic classes (non-BLS classes and enabled BLS classes) can still transmit frames. While I is blocked, its credit also decreases at a rate of s_I^I. Once I's credit reaches a certain resume level L_I, I is enabled again. Credit cannot fall below 0. Figure 1 shows an example. In Figure 1a frames of the green BLS class are sent back-to-back until its credit reaches H_I. Note that, due to non-preemptive link access, frame 3, which arrived just before H_I was reached, can still be sent. The credit, however, does not increase beyond H_I. Also note that, before frame 4 can be sent after credit replenishment, a lower-priority frame might have just started transmitting before the green class reached L_I. Figure 1b shows, how the green class' credit is reduced by higher-priority interference during its enable interval.

A. Formal Worst-Case Analysis of TSN/BLS

Here, we present a CPA-compatible local analysis to compute the worst-case transmission latency of a frame at a particular switch. A frame's transmission latency is defined to be the time from when it has been fully received at a switch's input port until it has been fully transmitted from an output port. In a switch, several delays contribute to the transmission latency. There is queueing delay at the input port and forwarding delay by the switch fabric. These two delays

[1]Some proposals for TSN/BLS suggest to drop frames of blocked BLS classes. Frame dropping, however, often entails retransmission of the dropped frames, which ultimately increases the number of frames being injected into the network. This, in turn, increases the probability of dropping even more frames. Others propose to temporarily change a BLS class' priority to the lowest one instead of blocking the class or dropping its frames. This, however, does not limit bursts from this class in the absence of lower-priority traffic [17].

are implementation dependent and are typically in the order of a few clock cycles, i.e. they are much smaller than the actual transmission latencies. Then there is transmission delay on the link, which is typically in the order of nanoseconds. Lastly, there is queueing delay at the output ports, which accounts for all delays induced by an output port's scheduler. As we model output ports as CPA resources, we focus on the output port's queueing delay during our analysis and assume that the first three delays only have negligible impact on the frame transmission latencies or can be modeled by constant terms.

As frame scheduling in Ethernet is non-preemptive, the worst-case transmission latency R_i^+ of a frame of stream i can be found by evaluating the worst-case queueing delay of all frame arrivals within its busy period [18].

Definition 1. *The worst-case queueing delay $w_i(q,a_i^q)$ of the q-th frame of a traffic stream i is the time interval from the instant the first frame of traffic stream i, which starts the level-i busy period [18], arrives at an output port until the q-th frame can be transmitted under the assumption that the q-th frame arrived at time a_i^q relative to the beginning of the busy period.*

We will explain the dependency of $w_i(q,a_i^q)$ on the arrival time a_i^q of the q-th frame later in this section. To calculate the worst-case queuing delay $w_i(q,a_i^q)$ for a frame at an output port's scheduler, we have to consider several blocking effects.

Shaper blocking: After the credit counter of BLS class $I \in \mathcal{B}$ has reached its upper limit H_I, I is blocked until its credit is reduced to at least its resume level L_I. This (potentially recurring) blocking time is called the *credit replenishment interval* of BLS class I.

Theorem 1. *The maximum credit replenishment interval of a BLS class $I \in \mathcal{B}$ is:*

$$t_I^{R+} = \left\lceil \frac{H_I - L_I}{-s_I^I} \right\rceil + \max_{j \in lp(I)} \{C_j^+\} \tag{2}$$

where $lp(I)$ yields the streams of all traffic classes with lower priority than class I.

Proof. The shaper credit of BLS class I is replenished (reduced) at rate s_I^I. The longest time for which frames of class I are blocked by I's shaper can be computed by dividing $H_I - L_I$ by I's idle slope (first term). However, as only streams of class I are blocked during this interval (other classes can still send frames), in the worst-case, the largest lower-priority frame might start its transmission just before I's credit reached L_I. This stretches the blocking time for frames of class I (second term), as shown by the LP frame in Figure 1a. □

Credit replenishment is required whenever the credit counter of BLS class I reaches H_I. The shortest time between any two replenishment intervals, i.e. the shortest time for I's credit to go from L_I to H_I, occurs when frames of class I are served back-to-back, i.e. without interruption by other frames. We call this time the shortest *service interval* of BLS class I.

Theorem 2. *The shortest service interval of a BLS class I during a busy period is:*

$$t_I^{S-} = \max\left\{ \left\lfloor \frac{H_I - L_I}{s_I^S} \right\rfloor, \min_{i \in I} \{C_i^+\} \right\} \tag{3}$$

Proof. First term: A blocked BLS class I is unblocked (enabled) as soon as its credit has been replenished, i.e. when I's credit just fell below L_I. Class I is blocked again the next time its credit reaches H_I. The credit difference $H_I - L_I$ is consumed at the fastest rate (i.e. in the shortest amount of time) when frames of class I are served back-to-back. This can be computed by dividing $H_I - L_I$ by I's send slope. *Second term:* If there are backlogged frames (which is the case during a busy period), then at least the smallest frame among all streams of class I receives service. Both terms are conservative lower bounds. Hence we can take their maximum. □

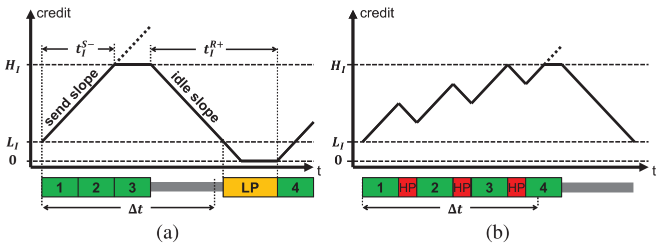

Fig. 1. TSN/BLS: (a) Frames of class I are sent at their maximum rate until H_I is reached and traffic from I is blocked for credit replenishment. (b) Frames of class I are interleaved with frames of other classes and class I's credit is decremented while these frames are being transmitted. Notice how in (b) more workload is released in the marked interval Δt.

Let $cl(i)$ be a function, which maps a traffic stream i to its corresponding traffic class and let Δw be the accumulated workload of BLS class $cl(i)$, i.e. the class stream i is assigned to. Then, the worst-case number of replenishment intervals required to serve Δw can be bounded by dividing Δw by $t_{cl(i)}^{S-}$. Consequently, the shaper blocking from class $cl(i)$ that is experienced by a stream i of this class is upper bounded by:

$$I_i^{SB}(\Delta w) = \left\lceil \frac{\Delta w}{t_{cl(i)}^{S-}} \right\rceil t_{cl(i)}^{R+} \tag{4}$$

Note that, traffic streams i of non-BLS traffic classes do not experience shaper blocking and $I_i^{SB}(\Delta w) = 0$ if $cl(i) \notin \mathcal{B}$.

Lower-priority blocking, in non-preemptive scheduling, occurs if the largest lower-priority frame starts transmitting just before the first frame of stream i can be transmitted. For BLS streams i, lower-priority blocking, is already included in the shaper blocking via Eq. (2). Non-BLS streams do not experience shaper blocking and we have to explicitly consider lower-priority blocking:

$$I_i^{LPB} = \max_{j \in lp(cl(i))} \{C_j^+\} \tag{5}$$

if $cl(i) \notin \mathcal{B}$. If $cl(i) \in \mathcal{B}$, we have $I_i^{LPB} = 0$, as discussed above.

Higher-priority blocking: Let $hp(i)$ be a function, which returns the set of all traffic streams with a priority higher than that of stream i. There are two cases of higher-priority blocking: (a) the higher-priority blocking is by frames of non-BLS streams $j \in hp(i) \wedge cl(j) \notin \mathcal{B}$ and (b) the higher-priority blocking is by frames of BLS streams $j \in hp(i) \wedge cl(j) \in \mathcal{B}$.

In case (a) the higher-priority blocking that a frame of stream i can experience in any time interval Δt can be bounded by summing all higher-priority frames, which arrive during Δt before the frame of stream i can be transmitted.

$$I_i^{HPB,nBLS}(\Delta t) = \sum_{j \in \{k | k \in hp(i) \wedge cl(k) \notin \mathcal{B}\}} \eta_j^+(\Delta t) C_j^+ \tag{6}$$

In case (b) we have to consider that the interference by higher-priority BLS traffic streams j of a BLS class J might be bounded by J's burst-limiting shaper. The shaper credit of a BLS class J is increased whenever frames of streams of J are transmitted and it is reduced during J's replenishment interval (i.e. after J's credit reached its limit H_J) or when, outside of J's replenishment interval, other-priority frames are interleaved with frames of streams of J. Selectively interleaving frames of J's traffic streams with frames of other-priority can potentially modify J's credit counter such that, in a given time interval Δt, J could release more interference on stream i (see Figure 1b) than without these interleaved frames (see Figure 1a). This implies that, in case (b), the critical instant scenario of CPA's local analysis might not be found by having frames of class J arrive as soon as possible (as it (correctly) is the case in Eq. (6) of case (a)). Instead, to generate the worst-case impact of BLS class J on stream i, we must allow J's frames to arrive in a way that allows the interleaving with frames of other traffic streams. Once we know how many frames of each stream of class J and of each potentially interleaving stream are available, we can formulate an integer linear program (ILP) to derive an interleaving pattern that maximizes the interference from BLS class J on stream i for any given time interval Δt:

$$\tilde{I}_{i,J}^{HPB,BLS}(\Delta t) = \text{maximize} \sum_{j\in J} x_j^J C_j^+ \tag{7}$$

where $x_j^J \in \mathbb{N}_{\geq 0}$ (recall that scheduling in Ethernet is non-preemptive) models how many frames of each stream $j \in J$ contribute to J's interference.

First, we define which interfering traffic streams of other priorities (classes) can be interleaved with class J. Since we are spanning the level-i busy period of stream i, a higher-priority BLS class J can be interleaved with frames of streams of any traffic class of higher or equal priority than class $cl(i)$. Let $D_i^J = hep(i) \setminus J$ be this set of interleaving streams, where $hep(i)$ yields all traffic streams with a priority higher than or equal to that of stream i (including i). Note that interleaved lower-priority frames are considered separately as part of J's interleaved replenishment intervals (see below).

We know that, at the end of each Δt, BLS class J's credit must be positive and below H_J, which can be modeled in the ILP by the following constraint:

$$0 \leq \sum_{j\in J} x_j^J s_J^S C_j^+ + \sum_{j\in D_i^J} x_j^D s_J^I C_j^+ + x^R s_J^I t_J^{R+} \leq H_J + s_J^S \max_{j\in J}\left\{C_j^+\right\} \tag{8}$$

where $x_j^D \in \mathbb{N}_{\geq 0}$ models how many frames of each interleaved traffic stream $j \in D_i^J$ are present. Eq. (8) also models, that J's credit can be reduced by inserting $x^R \in \mathbb{N}_{\geq 0}$ replenishment intervals t_J^{R+}. Each replenishment interval reduces J's shaper credit by $s_J^I t_J^{R+}$. Frames of streams of J are non-preemptive. Hence, in the worst-case, the largest frame of J can start transmitting just before J reaches H_J. Consequently, we extend the upper credit bound in Eq. (8) by $s_J^S \max_{j\in J}\{C_j^+\}$.

In any time interval Δt only a limited amount of workload (accumulated over all traffic classes) can be released by a port's scheduler, i.e. Δt plus one frame, which is released just before Δt ends [13]. This transitively also holds for BLS class J and its interleaving frames from D_i^J (as J and D_i^J generate a subset of this accumulated workload) and J's replenishment intervals (as we consider these intervals as blocking terms for frames of J). For J this amount of accumulated workload is at most Δt plus the longest frame among the streams of J (if this frame is released just before Δt ended). Hence, can we formulate a workload constraint for the ILP:

$$0 \leq \sum_{j\in J} x_j^J C_j^+ + \sum_{j\in D_i^J} x_j^D C_j^+ + x^R t_J^{R-} \leq \Delta t + \max_{j\in J}\{C_j^+\} \tag{9}$$

Here, we conservatively assume, that each replenishment interval only contributes with its shortest delay (cf. Theorem 2):

$$t_J^{R-} = \left\lfloor \frac{H_J - L_J}{-s_J^I} \right\rfloor \tag{10}$$

The number of frames of streams of J and interleaving streams are bounded by their event models, i.e. $0 \leq x_j^J \leq \eta_j^+(\Delta t), \forall j \in J$ and $0 \leq x_j^D \leq \eta_j^+(\Delta t), \forall j \in D_i^J$. The number of replenishments intervals can be bounded by $0 \leq x^R \leq \eta_J^{R+}(\Delta t)$. Where $\eta_J^{R+}(\Delta t)$ considers that, in the worst-case, the shortest service interval of BLS class J, t_J^{S-}, is always followed by the shortest credit replenishment interval of J:

$$\eta_J^{R+}(\Delta t) = \left\lceil \frac{\Delta t}{t_J^{S-} + t_J^{R-}} \right\rceil \tag{11}$$

The higher-priority blocking by BLS classes on a traffic stream i can be bounded by conservatively assuming that the interference from different BLS classes J is independent and by summing the interference over all these J.

$$I_i^{HPB,BLS}(\Delta t) = \sum_{J\in\{cl(j)|j\in hp(i)\}\cap\mathcal{B}} \tilde{I}_{i,J}^{HPB,BLS}(\Delta t) \tag{12}$$

Same-priority blocking: The q-th frame of stream i, which arrived at time a_i^q, can be blocked by $q-1$ previously released frames of its own stream and by all frames of same-priority streams, which have been queued prior to a_i^q:

$$I_i^{SPB}(q, a_i^q) = (q-1)C_i^+ + \sum_{j\in sp(i)} \eta_j^{+]}(a_i^q) C_j^+ \tag{13}$$

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

Ethernet serves frames of equal priority in FIFO order. In [12] it has been shown that FIFO scheduling requires a candidate search to compute the worst-case blocking (the earlier a frame arrives, the longer its transmission latency might be, and the later a frame arrives, the more blocking from previously queued frames it might experience). The set of arrival candidates a_i^q, which must be investigated for the q-th arrival of a frame of stream i, can be reduced to the times at which a_i^q coincides with frame arrivals of interfering same-priority traffic streams [12]:

$$A_i^q = \bigcup_{j\in sp(i)} \left\{\delta_j^-(n) | \delta_i^-(q) \leq \delta_j^-(n) < \delta_i^-(q+1)\right\}_{n\geq 1} \tag{14}$$

Finally, we can compute the worst-case queueing delay $w_i(q, a_i^q)$ for the q-th arrival of a frame of stream i, which occurred at time a_i^q:

$$\begin{aligned} w_i(q,a_i^q) = {} & I_i^{SB}(I_i^{SPB}(q,a_i^q) + C_i^+) + I_i^{LPB} + I_i^{SPB}(q,a_i^q) + \\ & I_i^{HPB,nBLS}(w_i(q,a_i^q)) + I_i^{HPB,BLS}(w_i(q,a_i^q)) \end{aligned} \tag{15}$$

Note that $I_i^{SB}(\Delta w)$ requires the accumulated workload of BLS class $cl(i)$ as its argument and $I_i^{SPB}(q, a_i^q)$ only considers $q-1$ frames of stream i (cf. Eq. (13)); hence the additional C_i^+. Also note that, as discussed above, $I_i^{SB}(\Delta w) = 0$ if $cl(i) \notin \mathcal{B}$ and $I_i^{LPB} = 0$ if $cl(i) \in \mathcal{B}$. Eq. (15) cannot be solved directly, since $w_i(q, a_i^q)$ occurs on both sides. In fact, it represents a fixed-point problem. As all terms are monotonically increasing, it can be solved by iteration. A valid starting point for the fixed-point iteration is, for example, $w_i(q, a_i^q) = (q-1)C_i^+$.

Following [13], the largest transmission latency $R_i(q)$ for the q-th arrival of a frame of stream i can computed by adding

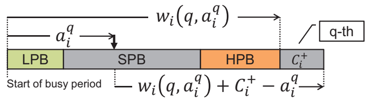

Fig. 2. Example queuing delay

the execution time of the q-th frame C_i^+ to its worst-case queueing delay and considering that the q-th frame arrived at time a_i^q (see Figure 2).

$$R_i(q) = \max_{a_i^q \in A_i^q} \{w_i(q,a_i^q) + C_i^+ - a_i^q\} \tag{16}$$

The worst-case frame transmission latency for a frame of stream i can be computed by taking the maximum of all $R_i(q)$:

$$R_i^+ = \max_{1 \le q \le \hat{q}_i} \{R_i(q)\} \tag{17}$$

where $\hat{q}_i$ is the maximum number of frame arrivals of stream i which have to be evaluated. Under 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 [18], which, in the TSN/BLS context, can be computed similar to the approach for Ethernet AVB in [13].

V. Experiments

In this section, we evaluate TSN/BLS by using our proposed analysis method and compare it to Ethernet AVB [12], [13] and IEEE 802.1Q [12]. We also implemented the optimization proposed by [13] in our TSN/BLS analysis, i.e. we account for the fact that an Ethernet link itself acts as an additional (physical) traffic shaper. Our evaluation is based on the quad star topology in Figure 3, which comprises four switches, each of which connects two electronic control units (ECUs) to the network. We use the worst-case end-to-end latency guarantees from our formal analysis as the comparison metric. The traffic in this network has been provided by Daimler AG and is summarized in Table I (detailed traffic cannot be disclosed). There are three traffic classes: control data traffic (CDT), general control traffic (GCT), and camera traffic (CAM). Latency-critical traffic is assigned to the CDT class. This class is mapped to the highest Ethernet priority and is shaped by TSN/BLS. GCT is assumed to be not as critical as CDT and is, hence, mapped to a priority level below CDT. High-bandwidth video traffic with comparatively relaxed latency requirements is assigned to the CAM class, which is mapped to a priority below GCT. The GCT and CAM traffic classes do not use TSN/BLS and are scheduled based on their priority. Traffic streams are either unicast, multicast, or broadcast. For multicast traffic, we use the notation $n(d)$ to indicate that there are n multicast streams to d destinations. For each traffic stream, Table I specifies periods and payloads with their average, minimum, and maximum values. The payload is given as the raw application payload. Since we assume that IPv4/UDP is used on the network and transport layers, 28bytes of protocol overhead must be added to this payload. We assume that ECUs inject traffic into the network according to a *periodic with jitter* event model [7] and that, in the worst-case, an ECU might inject an occasional burst of two frames into the network. Thus, we set the jitter to the period. The transmission delay of each Ethernet link is assumed to be 33ns, modeling wire length of 10m. Next, we evaluate the impact of traffic shaping in TSN/BLS and AVB on CDT traffic. Then, we investigate the impact of the shaped CDT traffic on the unshaped lower-priority GCT and CAM traffic.

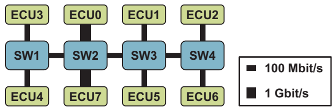

Fig. 3. Quad star topology

As suggested by [14], we set the idle slope of a BLS class I, per switch port, to its average bandwidth requirement: $s_I^I = -\min\{\alpha_I \sum_{i \in I}(C_i^+/P_i), r_{TX}\}$, where P_i is the period of traffic stream i. Additionally, we introduce an overreservation factor α_I to scale BLS class I's bandwidth requirement beyond its bare minimum, e.g. to prevent transient overload from creating permanent delay. Following [14], the send slope of a BLS class I is set to the remaining bandwidth: $s_I^S = r_{TX} + s_I^I$. To avoid setting the maximum credit H_I for each port explicitly, we derive it from class I's send slope to $H_I = s_I^S \Gamma_I$, so that (the time interval) Γ_I has an almost direct impact on the shortest service interval of BLS class I (cf. Eq. (3) for sufficiently small L_I). L_I is set to a certain percentage of H_I. AVB's shapers are configured as in [12].

We evaluate TSN/BLS by considering 16 configurations, which follow the naming scheme *BLS* Γ_I/α_I. For Γ_I, we evaluate durations of *2.5us*, *5us*, *10us*, and *25us*. We assume that the shaper's resume level is 5% of H_I. For the overreservation α_I, we chose values of *2*, *4*, *8*, and *16*. For the comparison with Ethernet AVB, we consider the AVB configurations *AVB 2*, *AVB 4*, *AVB 8*, and *AVB 16* with the same α_I as for TSN/BLS. We assume that only CDT traffic is shaped by AVB's CBS.

Figure 4 shows the worst-case end-to-end latency guarantees from our formal TSN/BLS analysis against the guarantees of AVB and IEEE 802.1Q. As we consider each path in the multicast and broadcast trees to be an individual path, we need to compare the latency guarantees of 94 paths. Although this is not a random experiment, we use boxplots to efficiently summarize these latency guarantees in a single box. The actual box covers 50% of the latency guarantees of all paths with its lower and upper edges representing the 25% and 75% quartiles. The whiskers indicate the minimum and maximum latency guarantees among the guarantees of all 94 paths. The median is marked by a red bar and the average by a black dot.

To understand Figure 4, we discuss the general impact of the parameters Γ_I and α_I on the shaper blocking, which is the main source of delay for CDT traffic. First, the average bandwidth requirement of CDT traffic is usually comparatively small compared to the link bandwidth ($\sim$1% on our setup), i.e. $s_I^I \ll r_{TX}$. Thus, while scaling by overreservation α_I has a direct impact on s_I^I, it only has a very limited impact on $s_I^S = r_{TX} + s_I^I$. The shaper blocking (Eq. (4)) mainly depends on t^{R+} (Eq. (2)) and t^{S-} (Eq. (3)). As can be seen from Eq. (2), a larger α_I, leads (via s_I^I) to a t^{R+}, which is by a factor α_I smaller (when we neglect the lower-priority blocking for the sake of looking at the broader picture). Since α_I's impact on s_I^S is limited, t^{S-} does not change significantly.

TABLE I
Traffic characteristics

	CDT	GCT	CAM
Unicast (#)	1	9	4
Multicast (#)	2(2)	6(2), 4(3), 2(4)	1(2)
Broadcast (#)	0	6	0
Payload (bytes)	[16, 380]	[3, 500]	[875, 1400]
Average (bytes)	200	74	1160
Period	5ms	[10ms, 1s]	[100us, 1ms]
Average	5ms	197ms	372us

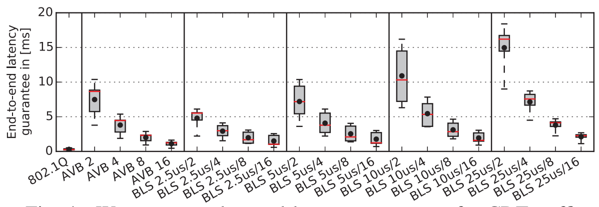

Fig. 4. Worst-case end-to-end latency guarantees for CDT traffic

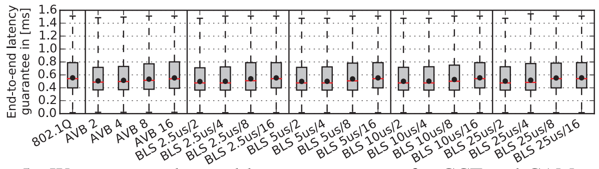

Fig. 5. Worst-case end-to-end latency guarantees for GCT and CAM traffic

Consequently, *the overreservation will mainly impact the replenishment interval*. Second, the credit limit $H_I = s_I^S \Gamma_I$ is mainly affected by Γ_I. It also depends on α_I (via s_I^S), but, as we have discussed, for typical CDT traffic, α_I's impact on s_I^S is very limited. From Eqs. (2) and (3), we see that *smaller values of H_I lead to* smaller values of t^{R+} and t^{S-}, i.e. *more but smaller replenishment intervals* (see Eq. (4)). We verified this impact of Γ_I and α_I on the shaper blocking by comparing t^{R+} and t^{S-} for different Γ_I and α_I during our analysis.

Figure 4 confirms these observations. For a given Γ_I, we see that increasing the overreservation α_I reduces (improves) the worst-case latency guarantees due to less shaper blocking. The main difference between AVB and TSN/BLS is the granularity of the credit replenishment. In AVB, credit replenishment is required on a per-frame basis, i.e. (conservatively assuming that there is no credit at the beginning of the busy period) once a frame has been sent, replenishment is required until the next frame can be sent [12]. In TSN/BLS, in contrast, credit replenishment is not required until H_I is reached, i.e. potentially more frames can be sent without replenishment, but replenishment intervals are also longer. In the worst-case a frame arrives just after its class' replenishment started and the frame must wait until the replenishment interval is over. The granularity of TSN/BLS' credit replenishment can be controlled by Γ_I, e.g. smaller Γ_I lead to more but shorter replenishment intervals. Having more frequent but shorter replenishment intervals can reduce the blocking time of individual frames (less waiting time once H_I is reached). This leads to smaller worst-case transmission latencies, which lead to smaller jitter values, which, in turn, cause less transient (over)load in the network. Hence, the overall worst-case latency guarantees improve (are lower) for smaller values of Γ_I. We can see that, with the right combination of Γ_I and α_I, TSN/BLS can achieve comparable and even lower (better) worst-case latency guarantees than AVB. In contrast to AVB and TSN/BLS, IEEE 802.1Q does not shape traffic and, hence, introduces no additional shaping delays. Consequently, IEEE 802.1Q gives the lowest worst-case latency guarantees (max. 0.5ms) among all configurations.

Figure 5 shows the end-to-end latency guarantees for (non-TSN) GCT and CAM traffic in the presence of CDT traffic. All configurations have roughly the same impact on this lower-priority traffic. Interestingly, comparing Figures 4 and 5, reveals that in some configurations the worst-case latency guarantees for *unshaped* lower-priority traffic are actually lower than the guarantees given by TSN/BLS for CDT traffic. In these cases it would, hence, make more sense to use IEEE 802.1Q than TSN/BLS. One argument for TSN/BLS, however, is its ability to restrain traffic within predefined bounds, e.g. to protect the network from malicious traffic or babbling idiots. This protection, however, could also be achieved by ingress filtering at the switches. This would separate filtering (protection) from scheduling. Such mechanisms are currently being discussed in the Ethernet TSN task group.

VI. Conclusion

We presented a formal analysis approach to derive worst-case timing guarantees for the burst-limiting shaper (BLS) of the upcoming Ethernet TSN standard. In contrast to related work, our analysis considers all blocking effects of this shaper, which enables a thorough timing analysis for all traffic streams. We used a typical automotive network to evaluate our analysis and compared the results against Ethernet AVB and standard Ethernet (IEEE 802.1Q). We also evaluated the impact of (shaped) TSN/BLS traffic on unshaped lower-priority traffic. Our experiments show that the worst-case guarantees that can be given by our approach for TSN/BLS suffer mainly from credit shaper replenishment. In fact, in our setup, unshaped (non-BLS) traffic sometimes had lower worst-case latency guarantees than the shaped (critical) BLS traffic. IEEE 802.1Q had the lowest worst-case latency guarantees.

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] Avionic Systems Group AS-2D2, "TTEthernet (SAE AS6802)," http://standards.sae.org/as6802/.

[4] D. Thiele, R. Ernst, and J. Diemer, "Formal Worst-Case Timing Analysis of Ethernet TSN's Time-Aware and Peristaltic Shapers," in *IEEE Vehicular Networking Conference (VNC)*, December 2015.

[5] 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.

[6] F.-J. Götz, "Alternative Shaper for Scheduled Traffic in Time Sensitive Networks," in *IEEE 802.1 TSN TG Meeting*, Vancouver, BC, Canada, January 2013.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] F. Reimann, S. Graf, F. Streit, M. Glas, and J. Teich, "Timing analysis of Ethernet AVB-based automotive E/E architectures," in *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2013.

[12] 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.

[13] 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.

[14] S. Kerschbaum, F.-J. Götz, and F. Chen, "Towards the Calculation of Performance Guarantees for BLS in Time-Sensitive Networks," in *IEEE 802.1 TSN TG Meeting*, Dallas, TX, USA, November 2013.

[15] 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.

[16] 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.

[17] C. Boiger, "How Many Transmission Selection Algorithms Do We Need?" in *IEEE 802.1 Interim Meeting*, Victoria, BC, Canada, May 2013.

[18] R. I. Davis and A. Burns, "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised," *Real-Time Systems*, vol. 35, 2007.