DOI: pending

Formal Worst-Case Performance Analysis of Time-Sensitive Ethernet with Frame Preemption

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

***Abstract*—One of the key challenges in future Ethernet-based automotive and industrial networks is the low-latency transport of time-critical data. To date, Ethernet frames are sent non-preemptively. This introduces a major source of delay, as, in the worst-case, a latency-critical frame might be blocked by a frame of lower priority, which started transmission just before the latency-critical frame. The upcoming IEEE 802.3br standard will introduce Ethernet frame preemption to address this problem. While high-priority traffic benefits from preemption, lower-priority (yet still latency-sensitive) traffic experiences a certain overhead, impacting its timing behavior. In this paper, we present a formal timing analysis for Ethernet to derive worst-case latency bounds under preemption. We use a realistic automotive Ethernet setup to analyze the worst-case performance of standard Ethernet and Ethernet TSN under preemption and also compare our results to non-preemptive implementations of these standards.**

I. Introduction

Future in-vehicle backbone networks will be based on Ethernet, as legacy buses such as CAN or FlexRay cannot keep pace with the growing bandwidth demands of advanced driver assistance systems (ADAS) or infotainment systems. Ethernet enables a scalable high-speed communication infrastructure and allows to realize arbitrary network topologies. However, Ethernet has a complex timing behavior. Thus, if safety-critical real-time applications are added to Ethernet, a formal analysis is required to verify that their timing requirements are met.

As an in-vehicle backbone network, Ethernet must be able to transport traffic streams of mixed-criticality. Standard Ethernet (IEEE 802.1Q) addresses this problem by introducing eight prioritized traffic classes. As there are typically more traffic streams than traffic classes in a network, streams must share traffic classes. In IEEE 802.1Q, arbitration between these traffic classes follows a static-priority-based scheme. Arbitration between frames of the same traffic class is usually done in FIFO order. Ethernet AVB and the upcoming Ethernet TSN (IEEE 802.1Qbv) [1] introduce additional traffic shapers on top of IEEE 802.1Q, e.g. in order to prevent starvation of lower-priorities or to implement time-triggered transmission of latency-critical traffic (respectively).

Until now, however, frame transmission in Ethernet is non-preemptive, regardless of the arbitration mechanism. Under non-preemptive frame transmission, a frame, which is in transmission, is guaranteed to finish without interruption. Hence, a time-critical high-priority frame can be delayed by a frame of lower-priority, if the high-priority frame arrives during the transmission of the lower-priority one. For instance, in the worst-case, high-priority frames can be delayed by about 120 us per switch at 100 MBit/s links by lower-priority frames. This might be too much for time-critical control applications.

978-1-5090-1314-2/16/$31.00 ©2016 IEEE

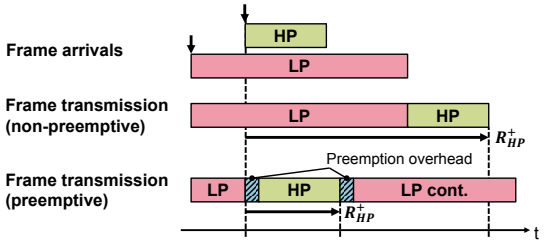


Fig. 1. Example of non-preemptive and preemptive frame transmission

The upcoming IEEE 802.3br standard addresses this problem by introducing frame preemption to Ethernet. With frame preemption the maximum delay that can be experienced by a higher-priority frame due to lower-priority blocking is reduced significantly to about 12 us per switch at 100 MBit/s. Frame preemption, however, also causes a certain overhead for preempted frames, which can have a measurable negative performance impact. This can become a problem when critical traffic streams with large frames, which are not scheduled on the highest priority, are preempted. ADAS (camera) traffic, which is usually scheduled on a priority below high-priority control traffic, is a typical example.

Figure 1 illustrates frame preemption in standard Ethernet inside a switch port. There are two frames: *HP* and *LP*, where *HP* has a higher priority than *LP*. Frame *HP* arrives while *LP* is in transmission. In the non-preemptive scenario, *HP* has to wait for *LP* to finish its transmission before it can be sent. This results in a long transmission latency R^+_{HP} of *HP*. Under frame preemption, the transmission of frame *LP* can be preempted to allow the higher-priority frame *HP* to be sent early. As preemption entails a certain overhead, i.e. the first fragment of frame *LP* must be terminated with a CRC followed by an inter frame gap, *HP* cannot be sent immediately. After *HP* has been transmitted, *LP* is resumed. Again, this introduces a certain overhead by prepending the new fragment of *LP* with a preamble and other information, which is required to reassemble the preempted frame at the receiving end. As can be seen, preemption decreases the delay that *HP* experiences.

DOI: pending

Observe, however, that the preemption overhead increases the time during which the switch port is busy transmitting data.

The **contribution of this paper** is twofold. First, we present a formal worst-case timing analysis for frame preemption in standard Ethernet and Ethernet TSN. Then, we evaluate the effect of frame preemption on performance by comparing the worst-case latency guarantees of our analysis with the results of a non-preemptive analysis in a realistic automotive setup.

II. Related Work

Simulation has traditionally been used to evaluate network performance. However, it typically entails long simulation runs. Furthermore, there is no guarantee that simulation exposes all corner cases, rendering it unsuitable to verify the timing behavior of timing- and safety-critical systems (e.g. highly-automated and autonomous driving). It has been shown, on the other hand, that compositional performance analysis (CPA) [2], real-time calculus (RTC) [3], or the trajectory approach [4] can be used to derive safe latency bounds for Ethernet networks. An RTC-based Ethernet analysis is given in [5]. AFDX, an avionics Ethernet implementation, is analysed in [4]. Formal Ethernet timing analyses based on CPA are [6] for IEEE 802.1Q, [7] for Ethernet AVB, and [8] for Ethernet TSN. To the best of our knowledge, no *formal* analysis to derive worst-case performance guarantees under IEEE 802.3br frame preemption has been proposed so far.

A simulation-based evaluation of IEEE 802.3br is presented in [9]. In [10], an experiment-based evaluation of a custom preemption mechanism, which has a slightly larger overhead than IEEE 802.3br, is presented. Both [9] and [10] confirm that frame preemption reduces the latency and jitter of high-priority traffic. However, no formal worst-case guarantees are given.

An early academic version of TTEthernet supported frame preemption. However, preempted frames were not resumed but retransmitted, resulting in poor link utilization. Recent commercial TTEthernet implementations [11] only support timely blocking (block non-critical frames, so that there is no overlap with time-triggered critical frames) and shuffling (accept delay from non-preemptiveness). In [12], an actual frame preemption mechanism for TTEthernet is presented. The evaluation, however, is only simulation-based and does neither consider latency nor jitter. Only frame drop rates are investigated.

III. Frame Preemption in Ethernet (IEEE 802.3br)

Frame preemption in Ethernet is specified in the IEEE 802.3br (interspersing express traffic) standard [13]. In the context of Ethernet TSN, IEEE 802.1Qbu (frame preemption) [14] adds management and configuration mechanisms for frame preemption. As we are interested in the timing impact of the actual preemption mechanism, we will focus on IEEE 802.3br.

IEEE 802.3br only allows one level of preemption. To this end, it defines two MAC interfaces: the *express* MAC interface and the *preemptable* MAC interface. Each Ethernet traffic class is mapped to either the express or the preemptable MAC interface. Frames of express classes cannot be preempted. Only frames of classes which are mapped to the preemptable MAC interface may be preempted by express frames. Particularly, preemptable frames cannot be preempted by other preemptable frames regardless of their priority. Note that IEEE 802.3br defines link-level frame preemption, i.e. frames are split into fragments and are reassembled at the MAC interfaces, so that switches (internally) only process complete frames.

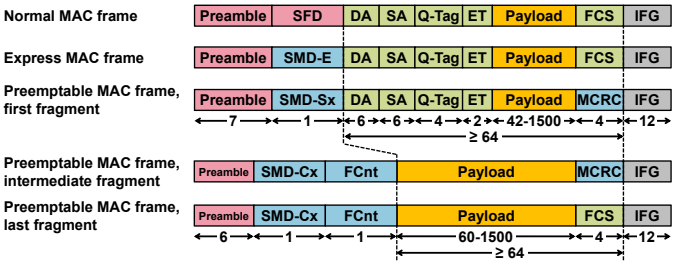


Fig. 2. MAC frame formats: original IEEE 802.3 MAC frame format on the top followed by four IEEE 802.3br frame formats (all sizes in bytes)

As a result of preemption, a frame is split into two or more fragments. One design goal of IEEE 802.3br was to preserve the basic structure of IEEE 802.3's MAC frame format in order to make frame preemption transparent to Ethernet's physical layer (PHY). Hence, each fragment must appear to the PHY as a valid Ethernet frame. To this end, IEEE 802.3br defines different MAC frame formats (see Figure 2). All formats start with a preamble and end with a CRC sum (FCS or MCRC) followed by an inter frame gap (IFG). Additionally, a (normal) IEEE 802.3 MAC frame comprises a start of frame delimiter (SFD), destination and source addresses (DA and SA), an IEEE 802.1Q tag (Q-Tag), an EtherType field (ET), and finally the actual payload, e.g. IP and UDP or TCP data.

The MAC frames of IEEE 802.3br's dual MAC interface are also called MFrames (MAC merge frames). The express frame format is very similar to the IEEE 802.3 MAC frame format. The only difference is that the SFD is replaced by an SMD-E (start MFrame delimiter - express) field to signal the express frame. The first preemptable frame is also very similar to the express frame. Its start, however, is signaled by the SMD-Sx (SMD - start fragment) delimiter. If this frame is not preempted, it is terminated by an FCS followed by an IFG (not shown in Figure 2). A preemptable frame may be split into fragments. With the exception of the last one, each fragment is terminated by an MCRC (instead of an FCS) followed by an IFG. This MCRC protects the fragment's data and is necessary to let the fragment appear as a valid IEEE 802.3 MAC frame to the Ethernet PHY. In the last fragment, this MCRC is replaced by the FCS of the entire original frame, i.e. if it had been transmitted without preemption. All fragments following the first one are signaled by SMD-Cx (SMD - continuation fragment) and a fragment counter (FCnt). As IEEE 802.3br only allows one level of preemption, DA, SA, Q-Tag, and ET only have to be transmitted once. Thus, all fragments after the first one can accommodate a slightly larger payload. SMD-Sx/Cx and FCnt are implemented as modulo-4 counters to facilitate the detection of lost fragments.

All MAC frames (including fragments) must meet the minimum Ethernet frame size requirement, which is 84 bytes (including IFG, Figure 2). If the actual payload is too small to meet this requirement, the payload field is padded accord-

DOI: pending

ingly. However, preemption is not allowed to introduce any additional padding to the resulting fragments. This imposes *constraints* on the preemption instant and the fragmentation granularity: (a) A preemptable frame cannot be preempted until the fragment (of it) that is currently being transmitted fulfills the minimum frame size requirement. (b) A preemptable frame or its remaining data can only be fragmented further if the resulting fragments meet the minimum frame size requirement.

From a worst-case perspective it is essential to determine the longest frame or fragment, which can block an express frame, and the preemption overhead per fragment.

Lemma 1. *The longest lower-priority frame or fragment that can block an express frame is 143 bytes long.*

Proof. Two cases must be considered: (a) a lower-priority frame that cannot be preempted and (b) a frame's last fragment that cannot be preempted further. For both cases, we first determine the smallest frame or fragment that can still be preempted. (a) As we are interested in the smallest frame, only one preemption is possible. If we split a frame, both resulting fragments must fulfill the minimum Ethernet frame size requirement of 84 bytes (preemption is not allowed to introduce additional padding, v.s.). Hence, our goal is to minimize the combined payload $p{=}p_1+p_2$ of both frames, s.t. Preamble + SMD-Sx+DA+SA+Q-Tag+ET+p_1+MCRC+IFG$\geq$84 bytes (first fragment) and Preamble + SMD-Cx + FCnt + p_2 + FCS + IFG$\geq$84 bytes (second fragment) (see Figure 2). This yields $p_1{=}42$ and $p_2{=}60$ bytes. Hence, the smallest frame, which can still be preempted, must have a payload of $p{=}102$ bytes, which results in an overall frame size of 144 bytes, if Preamble, SMD-Sx, DA, SA, Q-Tag, ET, FCS/MCRC, and IFG are added. (b) Any remaining data can only be split further, if the resulting fragments fulfill the minimum Ethernet frame size requirement of 84 bytes. Here we minimize $p{=}p_1 + p_2$ s.t. Preamble + SMD-Cx + FCnt + p_1 + MCRC + IFG$\geq$84 bytes and Preamble + SMD-Cx + FCnt + p_2 + FCS + IFG$\geq$84 bytes, which yields $p_1{=}p_2{=}60$ bytes. Hence, the smallest amount of remaining data p that can still be preempted is 120 bytes and results in an overall fragment size of 144 bytes, if Preamble, SMD-Cx, FCnt, FCS, and IFG are added. In both cases, we get the largest frame/fragment that cannot be preempted further by subtracting 1 byte from the payload p, yielding frames/fragments of 143 bytes. □

Lemma 2. *The overhead per preemption is 24 bytes.*

Proof. As each additional fragment requires an additional Preamble, SMD-Cx, FCnt, MCRC, and IFG, the preemption overhead of a single preemption is 24 bytes (see Figure 2). □

IV. Compositional Performance Analysis (CPA)

In this paper, we use CPA to reason about the worst-case timing behavior of Ethernet [2] [15]. In CPA, abstract resources provide service according to some scheduling policy (e.g. IEEE 802.1Q or IEEE 802.1Qbv). For the Ethernet analysis, switch ports (as the points of arbitration) are modeled as resources [15]. An Ethernet traffic stream is defined to be a sequence of (related) Ethernet frames between a source and one (or more) destination(s). Such a stream is modeled as a chain of dependent frames, which are mapped to switch ports (CPA resources) according to its path through the network [15]. On each switch port, a frame consumes service according to its transmission time bounds C^- and C^+. These bounds are defined to be the shortest and longest time it takes to transmit the frame in the absence of any interference. For frames of a stream i the minimum and maximum frame transmission times depend on their minimum and maximum payload $p_i^{-/+}$:

$$C_i^{-/+} = \Big(42\,\text{bytes} + \max\{42\,\text{bytes}, p_i^{-/+}\}\Big)/r_{TX} \qquad (1)$$

where r_{TX} is the link speed. The constant terms model protocol overhead and minimum frame size (Section III). Potential preemption overhead will be considered during our analysis.

Frame arrivals (and emissions) are abstracted by event models [2]. In contrast to a single event trace, event models only consider the best- and worst-case behavior. This behavior is bounded by a pair of arrival functions $\eta^{-/+}(\Delta t)$, which yield the minimum and maximum number of frame arrivals in any half-open time interval $[t, t+\Delta t)$. Arrival functions have pseudo-inverse minimum distance functions $\delta^{+/-}(q)$, which yield the maximum and minimum time interval between the first and the last event in any sequence of q events.

CPA comprises two iterative analysis steps: A *local analysis*, which is explained in Sections V and VI, derives the best- and worst-case frame transmission latencies per switch port. From these latencies new output event models are derived [2]. The output event model of a frame on a switch port becomes its input event model on the next port. Event models are propagated by a *global analysis* loop, which ends if all event models become stable and initiates a new local analysis with the new event models otherwise [2]. Afterwards, worst-case end-to-end latency guarantees can be computed by summing the frame transmission latencies along a chain of frames.

V. Compositional Performance Analysis of Frame Preemption in Standard Ethernet (IEEE 802.1Q)

This section presents a local analysis, which considers the effects of frame preemption in IEEE 802.1Q. We focus on the computation of the worst-case frame transmission latencies for both frames of express and frames of preemptable traffic streams. The frame transmission latency is the time from the arrival of a frame at a switch input port until it has been fully sent from its output port. There are several delays, which a frame experiences while traversing a switch: queueing delay at its input port, forwarding delay in the switch fabric, queuing delay at its output port, and transmission delay on the link to the next switch. Since we only model the output ports of a switch as resources in CPA, we focus on the output queueing delay, which considers all delays from interfering traffic streams at an output port. The other delays are implementation dependent and typically in the order of a few clock cycles. Hence, they only have negligible impact on the transmission latency of a frame and can be modeled by a constant delay.

As we will see later, the transmission latency of a frame of traffic stream i can be derived from its queueing delay. To compute a frame's worst-case transmission latency, the worst-case queueing delays of all frame arrivals of stream i within

DOI: pending

its level-i busy period must be evaluated [16]. The level-i busy period is the longest period of time during which a switch port is busy processing frames of streams of priority i or higher. This also includes interference from other streams.

Strictly speaking, [16] only applies to non-preemptive scheduling. In the formal analysis context, the main difference between preemptive and non-preemptive scheduling is that in the former, preemption can occur at any time, while in the latter, once a non-preemptive entity (e.g. a frame) started transmitting, it cannot be interrupted. Hence, as discussed in Section III, IEEE 802.3br technically is still non-preemptive, but at a much smaller granularity (fragments), and the general approach from [16] remains applicable. Still, to support IEEE 802.3br frame preemption, we have to adjust the formal definition of the queueing delay from related work, e.g. [8].

We consider a switch output port and assume that the q-th frame of a traffic stream i arrives at time a_i^q relative to the beginning of stream i's level-i busy period, which was started by the arrival of the first of the q frames.

Definition 1. *The worst-case queueing delay $w_i(q,a_i^q)$ is the time interval from the beginning of stream i's level-i busy period until the last non-preemptable part of said q-th frame can be transmitted.*

For express traffic this *part* is the frame itself. For preemptable traffic it is a minimum-sized (non-preemptable) fragment.

We distinguish two sets of traffic classes. Let $\mathcal{P}$ be the set of *preemptable* traffic classes and $\mathcal{E}$ be the set of (non-preemptable) *express* traffic classes. Let the function $cl(i)$ map a traffic stream i to its traffic class. As discussed, there is only one level of preemption. Express frames cannot preempt each other regardless of their priority and also preemptable frames cannot preempt each other regardless of their priority. Only preemptable frames can be preempted by express frames. As express traffic is considered to be more critical than preemptable traffic, we assume that all express traffic streams have a higher Ethernet priority than preemptable streams.

In order to compute the worst-case queueing delay, we have to take into account several blocking effects.

Lower-priority blocking: *Preemptable frames* can only be preempted by higher-priority express frames. Consequently, the worst-case lower-priority blocking for a preemptable frame of traffic stream i occurs when the largest frame of a traffic stream with lower-priority than stream i starts transmitting right before the first frame of stream i starts transmitting [7]. Let $lp(i)$ yield the set of traffic streams of lower priority than that of stream i. Then we have for the lower-priority blocking of preemptable frames:

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

The lower-priority blocking for *express traffic* can potentially be reduced by taking frame preemption into account. Hence, in the following, we distinguish between $lp^E(i)$, the set of express traffic streams of lower priority than stream i and $lp^P(i)$, the set of preemptable traffic streams of lower priority than stream i.

Theorem 1. *The longest lower-priority blocking for frames of express traffic stream i is*

$$I_i^{LPB,E} = \max\Big\{ \underbrace{\max_{j \in lp^E(i)} \{C_j^+\}}_{(a)}, \min\Big\{ \underbrace{\max_{j \in lp^P(i)} \{C_j^+\}}_{(b)}, \underbrace{\tfrac{143\,\text{bytes}}{r_{TX}}}_{(c)} \Big\} \Big\} \tag{3}$$

Proof. As express frames cannot preempt each other, the longest blocking time from lower-priority express frames is given by term (a). The longest blocking time from lower-priority preemptable frames can be derived from the longest non-preemptable fragment. This fragment has a size of 143 bytes (Lemma 1), which can be translated to a blocking time by dividing by the link speed r_{TX} in term (c). If all lower-priority preemptable frames are shorter than 143 bytes, this can be exploited by taking the minimum of the largest lower-priority preemptable frame (term (b)) and term (c). We do not know which term (term (a) or the minimum over terms (b) and (c)) is larger. Thus, we take the maximum. □

Same-priority blocking: A frame of traffic stream i can be blocked by frames from traffic streams of equal priority [7]. Let $sp(i)$ yield all traffic streams with a priority equal to that of stream i (excluding stream i). For *express traffic*, the q-th frame of stream i, which arrived at time a_i^q, must wait until all frames of other same-priority streams, which arrived before frame q, and its own $q-1$ predecessors have been transmitted before it can be sent. If interfering frames arrive concurrently with frame q at a_i^q, we assume the worst-case ordering. To this end, we define the event arrival function $\eta^{+]}(\Delta t)$, which yields the number of frame arrivals in any *closed* time interval $[t,t+\Delta t]$. The same-priority blocking can be be computed as:

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

For *preemptable traffic*, only the last non-preemptable fragment is not part of the queueing delay (Definition 1). Thus, in addition to all frames of other same-priority streams, which arrived before q, and its own $q-1$ predecessors, the last fragment of q must also wait for all its preceding fragments. To maximize the potential number of preemptions of q, its last fragment is, in the worst-case, of minimum size (i.e. 84 bytes), leaving the rest of q's size ($C_i^+ - {}^{84\,\text{bytes}}/_{r_{TX}}$) for preemption.

$$I_i^{SPB,P}(q,a_i^q) = (q-1)C_i^+ + C_i^+ - \tfrac{84\,\text{bytes}}{r_{TX}} + \sum_{j \in sp(i)} \eta_j^{+]}(a_i^q) C_j^+ \tag{5}$$

In Ethernet, frames of equal priority are usually processed in FIFO order. To compute the worst-case blocking for the q-th frame of stream i under FIFO scheduling, a candidate search is required [17]: The earlier frame q arrives (within its jitter bounds), the longer its transmission latency might be (e.g. by blocking from (some of) its own $q-1$ queued predecessors). The later it arrives (within its jitter bounds), the more blocking from other same-priority frames, which have been queued before its arrival, it might experience. According to [17], all candidates a_i^q for the arrival time of the q-th frame of stream i coincide with the arrivals of interfering same-priority frames:

$$A_i^q = \bigcup_{j \in sp(i)} \left\{ \delta_j^-(n) | \delta_i^-(q) \le \delta_j^-(n) < \delta_i^-(q+1) \right\}_{n \ge 1} \tag{6}$$

DOI: pending

Higher-priority blocking: In any time interval Δt, a frame of traffic stream i can be blocked by all higher-priority frames, which arrive before this frame can be transmitted [7]. Let $hp(i)$ yield the set of all traffic streams with a priority higher than that of stream i. Then we have for the higher-priority blocking:

$$I_i^{HPB}(\Delta t) = \sum_{j \in hp(i)} \eta_j^+(\Delta t) C_j^+ \tag{7}$$

If this higher-priority blocking is due to preemption, there is a certain overhead. This overhead is considered separately (see below).

Preemption overhead: In IEEE 802.3br, only preemptable frames may experience preemption overhead. If the number of preemptions within a frame's queueing delay is known, their overhead can be modeled as an additional blocking term, extending the queueing delay. We start by deriving an upper bound on the maximum number of preemptions per frame.

Lemma 3. *The maximum number of preemptions of a single frame of traffic stream i is given by*

$$F_i^+ = \left\lfloor \frac{p_i^+ - 42\,\text{bytes}}{60\,\text{bytes}} \right\rfloor \tag{8}$$

Proof. The number of preemptions of a frame of stream i is maximized, if we distribute the frame's maximum payload p_i^+ among as many fragments as possible. In IEEE 802.3br, the minimum payload of the first fragment of a preempted frame is 42 bytes (see Figure 2). All following (non-initial) fragments must carry a minimum payload of 60 bytes (see Figure 2). Thus, we can compute the maximum number of preemptions of a single frame by first subtracting 42 bytes from its maximum payload p_i^+ and then dividing the remaining payload by 60 bytes. If $p_i^+ - 42$ is not divisible by 60 without remainder, one of the fragments must be larger and also carry the otherwise remaining $x < 60$ bytes, as IEEE 802.3br does not allow fragments smaller than the minimum Ethernet frame size requirement (see Section III). Hence, we round down. □

The maximum number of frame preemptions in the queueing delay can be computed by multiplying the number of preemptable frames in this period by their respective F_i^+.

A preemptable frame of traffic stream i can be blocked by at most one lower-priority preemptable frame (cf. Eq. (2)). As preemptable frames cannot preempt each other, this lower-priority frame will be transmitted entirely, but may be fragmented due to preemptions from express traffic. The maximum number of preemptions of such a lower-priority frame is thus:

$$N_i^{LP} = \max_{j \in lp^P(i)} \left\{ F_j^+ \right\} \tag{9}$$

Akin to the same-priority blocking in Eq. (5), we can compute the maximum number of frames of preemptable traffic streams with identical priority as stream i. If we consider the arrival of the q-th frame of stream i, which arrived at time a_i^q (within its busy period), the maximum number of preemptions from frames of same-priority traffic streams (including i) is:

$$N_i^{SP}(q, a_i^q) = qF_i^+ - 1 + \sum_{j \in sp(i)} \eta_j^{+]}(a_i^q) F_j^+ \tag{10}$$

Similar to Eq. (5), we do not consider the preemption overhead of the last fragment. This fragment (and its preemption overhead) will be considered later when we derive the worst-case transmission latency of the q-th frame of traffic stream i. $N_i^{SP}(q, a_i^q)$ is subject to the same candidate search as Eq. (5).

Within any time interval of length Δt, the number of preemptable frames of higher priority than stream i can be computed by summing up the higher-priority frame arrivals over Δt (cf. Eq. (7)). Let $hp^P(i)$ be this set of preemptable traffic streams of higher priority than that of stream i. The maximum number of frame preemptions from these higher-priority frames can then be computed:

$$N_i^{HP}(\Delta t) = \sum_{j \in hp^P(i)} \eta_j^+(\Delta t) F_j^+ \tag{11}$$

Equations (9), (10), and (11) bound the maximum number of frame preemptions a frame of traffic stream i can experience during its queueing delay by assuming that all frames are split into their maximum number of fragments. However, the number of frame preemptions can also be bounded by the actual (worst-case) preemption pattern from higher-priority express traffic. Let $hp^E(i)$ be this set of express traffic streams of higher priority than stream i. The following theorem computes a bound on the preemption overhead considering both bounds.

Theorem 2. *The preemption overhead in a time interval of length Δt containing q frames of the traffic stream under analysis i, out of which the q-th one arrived at time a_i^q relative to the beginning of Δt, is upper bounded by:*

$$I_i^{PO}(\Delta t, q, a_i^q) = \frac{24\,\text{bytes}}{r_{TX}} \min \left\{ \sum_{j \in hp^E(i)} \eta_j^+(\Delta t), N_i^{LP} + N_i^{SP}(q, a_i^q) + N_i^{HP}(\Delta t) \right\} \tag{12}$$

Proof. From Lemma 2 we know that the preemption overhead is 24 bytes. Its duration can be derived by dividing it by the link speed r_{TX}. Given Δt, q, and a_i^q, the maximum number of frame preemptions can be computed by adding Eqs. (9), (10), and (11) (second term in minimum). In the worst-case each higher-priority express frame causes a preemption. Hence, there can be no more frame preemptions than there are higher-priority express frames within Δt (first term in minimum). So, we take the minimum of both terms. □

For express and preemptable traffic, the worst-case queueing delay, can be derived by considering their respective blocking terms. For *express traffic streams* i, the worst-case queueing delay for the q-th frame of i, which arrived at time a_i^q, is:

$$w_i^E(q, a_i^q) = I_i^{LPB,E} + I_i^{SPB,E}(q, a_i^q) + I_i^{HPB}(w_i^E(q, a_i^q)) \tag{13}$$

As $w_i^E(q, a_i^q)$ occurs on both sides, Eq. (13) cannot be solved directly. It represents a fixed-point problem (i.e. the blocking terms increase $w_i^E(q, a_i^q)$, which, in turn, might lead to even more blocking) and can be solved by iteration, as all terms are monotonically increasing [2] [7]. A starting point is $(q-1)C_i^+$.

DOI: pending

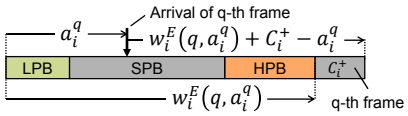


Fig. 3. Frame transmission latency computation example (cf. [8])

For *preemptable traffic streams* i, we have to consider the additional preemption overhead when computing the worst-case queueing delay $w_i^P(q,a_i^q)$:

$$w_i^P(q,a_i^q)=I_i^{LPB,P}+I_i^{SPB,P}(q,a_i^q)+I_i^{HPB}(w_i^P(q,a_i^q))+ I_i^{PO}(w_i^P(q,a_i^q),q,a_i^q) \quad (14)$$

Like Eq. (13), Eq. (14) also represents a fixed-point problem. A valid starting point for the iterative solution is $(q-1)C_i^+$.

From the queueing delay, we can derive the worst-case transmission latency $R_i(q)$ of the q-th frame of stream i by evaluating all of its arrival candidates $a_i^q \in A_i^q$. As the queueing delay is defined to be the time until the last non-preemptable part (frame or fragment) can be sent, we have to consider this last part when deriving the worst-case transmission latency. For express frames the last non-preemptable part is its transmission time C_i^+. For preemptable frames it is the smallest fragment size (84 bytes, already including 24 bytes of preemption overhead), assuming that, in the worst-case, their remaining payload has been used during the computation of the queueing delay (Eq. (5)). Figure 3 shows an example for express frames.

$$R_i(q)=\begin{cases}\max\limits_{a_i^q\in A_i^q}\left\{w_i^E(q,a_i^q)+C_i^+-a_i^q\right\} & \text{if } cl(i)\in\mathcal{E}\\ \max\limits_{a_i^q\in A_i^q}\left\{w_i^P(q,a_i^q)+\frac{84\,\text{bytes}}{r_{TX}}-a_i^q\right\} & \text{if } cl(i)\in\mathcal{P}\end{cases} \quad (15)$$

The worst-case frame transmission latency R_i^+ over all frames of stream i can be found by taking the maximum of all $R_i(q)$, $\forall q \le \hat{q}_i$. Where $\hat{q}_i$ is the maximum number of frame arrivals of stream i, which must be evaluated and, according to [16], equals the maximum number of frame arrivals of stream i in the longest level-i busy period.

$$R_i^+=\max_{1\le q\le \hat{q}_i}\{R_i(q)\} \quad (16)$$

In the remainder of this section, we present how to compute the longest level-i busy periods from which $\hat{q}_i$ can be derived. By definition, when computing the level-i busy period, there is no need to distinguish individual frame arrivals q nor their arrival times a_i^q. Thus, for express and preemptable traffic the same-priority blocking in Eqs. (4) and (5) can be replaced by:

$$\hat{I}_i^{SPB}(\Delta t)=\sum_{j\in sp(i)\cup\{i\}}\eta_j^+(\Delta t)C_j^+ \quad (17)$$

Likewise, the preemption overhead $I_i^{PO}(\Delta t,q,a_i^q)$ (Eq. 12) is replaced by $\hat{I}_i^{PO}(\Delta t)$, which is defined just like $I_i^{PO}(\Delta t,q,a_i^q)$, but instead of $N_i^{SP}(q,a_i^q)$, it uses:

$$\hat{N}_i^{SP}(\Delta t)=\sum_{j\in sp(i)\cup\{i\}}\eta_j^+(\Delta t)F_j^+ \quad (18)$$

Now, for express and preemptable streams the level-i busy periods $\hat{w}_i^E$ and $\hat{w}_i^P$ can be computed by:

$$\hat{w}_i^E=I_i^{LPB,E}+\hat{I}_i^{SPB}(\hat{w}_i^E)+I_i^{HPB}(\hat{w}_i^E) \quad (19)$$

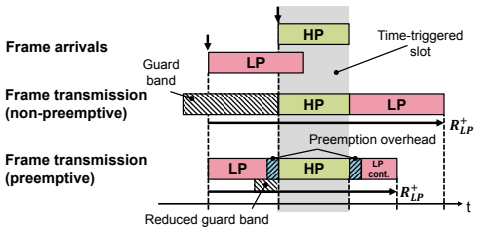


Fig. 4. Example of non-preemptive and preemptive frame transmission in IEEE 802.1Qbv

and

$$\hat{w}_i^P=I_i^{LPB,P}+\hat{I}_i^{SPB}(\hat{w}_i^P)+I_i^{HPB}(\hat{w}_i^P)+\hat{I}_i^{PO}(\hat{w}_i^P) \quad (20)$$

Eqs. (19) and (20) represent fixed-point problems, which, again, can be solved by iteration. A valid starting point is e.g. C_i^+. Then, the maximum number of frame arrivals of traffic stream i in the longest level-i busy period is $\hat{q}_i=\eta_i^+(\hat{w}_i^E)$ for express traffic and $\hat{q}_i=\eta_i^+(\hat{w}_i^P)$ for preemptable traffic.

VI. Compositional Performance Analysis of Frame Preemption in Ethernet TSN (IEEE 802.1Qbv)

Ethernet TSN's time-aware shaper (IEEE 802.1Qbv) [1] introduces time-triggered frame forwarding to Ethernet. It specifies (periodic) time-triggered slots in which traffic from (highly) critical traffic classes is scheduled without interference from other traffic classes at predefined points in time. Traffic is categorized according to the priority of its Ethernet traffic class. Typically, each critical traffic class has its own time-triggered slot and is only forwarded in this slot [1]. Outside these slots, traffic from all other (less- or non-critical) classes is scheduled according to its class' priority.

Originally, frame transmission in IEEE 802.1Qbv is non-preemptive. To protect the time-triggered slots from less-critical traffic extending into these slots and delaying critical traffic, e.g. when a less-critical frame started transmission just before the beginning of a slot, IEEE 802.1Qbv introduces *guard bands*. These are time intervals, which are inserted before the beginning of each time-triggered slot and during which no frame transmission is allowed to start. So, to protect the time-triggered slots, the length of each guard band must at least be equal to the transmission time of the longest non-time-triggered frame. This can result in poor link utilization.

As the intention of preemption is to reduce the latency of critical traffic, we assume that time-triggered traffic is treated as express traffic, implying that frames of time-triggered streams cannot be preempted. As IEEE 802.1Qbv already guarantees their forwarding free from interference from other traffic classes as soon as their slot starts, they do not benefit from the preemption of other traffic. As proposed by [18], less-critical traffic is assigned to the set of preemptable classes[1]. Under frame preemption, the performance of this less-critical traffic can be improved, as now the guard bands can be reduced to the maximum non-preemptable fragment length (Lemma 1).

This is illustrated in Figure 4. Without preemption, the less-critical frame *LP*, which arrives during its guard band,

[1]Note that our approach can be extended to cover cases where (some) less-critical traffic streams are also treated as express traffic.

DOI: pending

is blocked to prevent overlap with the time-triggered slot of the *HP* frame. Only after *HP*'s slot is over, *LP* can transmit. With preemption, however, the guard band is smaller, so that in this particular example *LP* can transmit immediately after its arrival. *LP* can transmit until it is preempted by *HP*'s slot. It resumes transmission after the slot is over, leading to a shorter transmission latency R^+_{LP} than in the non-preemptive scenario. Note that, when computing worst-case bounds, this reduced guard band must be considered unusable by *LP* and is modeled as an additional blocking term (see Theorem 3 below).

In [8], a CPA-compatible formal performance analysis for non-preemptive IEEE 802.1Qbv is presented. As critical time-triggered traffic is not affected by preemption, we *focus on extending this analysis to support preemption of less-critical (non-time-triggered) traffic*. Since frames of less-critical traffic streams (outside the time-triggered slots) use the preemptable MAC interface, they cannot preempt each other regardless of their priority. They can only be preempted by IEEE 802.1Qbv's scheduler, i.e. to schedule time-triggered slots. Hence, **lower-**, **same-**, and **higher-priority blocking** can be modeled as in IEEE 802.1Q [8], i.e. by Eqs. (2), (5), and (7), while taking into account that only less-critical streams (from $\mathcal{P}$) must be considered for higher-priority blocking in Eq. (7) [8]. Also, [8] shows that the interference from time-triggered slots (of critical traffic) can be modeled by (periodic) blocking terms.

Blocking by critical traffic: We start by bounding the maximum interference a single time-triggered slot can cause.

Theorem 3. *The maximum blocking caused by a single time-triggered slot from a class $J \in \mathcal{E}$ of length t_J^{TT} on a non-time-triggered traffic class $I \in \mathcal{P}$ is*

$$\tilde{t}^{TT}_{I,J} = \min\left\{ \max_{i \in \bigcup_{I \in \mathcal{P}} I} \{C_i^+\}, \frac{143\,\text{bytes}}{r_{TX}} \right\} + t_J^{TT} \tag{21}$$

Proof. The proof is similar to [8]. The minimum term models the minimum guard band length required to protect the time-triggered slot of class J from overlapping less-critical traffic. This length is either the maximum non-preemptable fragment length $^{143\,\text{bytes}}/_{r_{TX}}$ (Lemma 1) or, the maximum over all frames of less-critical traffic streams, if all these frames are smaller than $^{143\,\text{bytes}}/_{r_{TX}}$. We do not know in which order the frames of streams of class I are processed. Thus, in the worst-case, we have to assume that non-time-triggered traffic arrives such that a frame or fragment of the length of the guard band becomes ready to transmit just after J's guard band started and that the guard band cannot be used to transmit any traffic of I. The second term is the length of the time-triggered slot of J. □

Let the period of the time-triggered slot of a traffic class $J \in \mathcal{E}$ be t_J^{CYC}. In any time interval Δt, the maximum number of times frames of a less-critical stream i can be blocked by traffic streams from a critical time-triggered traffic class J can be computed by dividing Δt by t_J^{CYC}. Conservatively assuming that, in the worst-case, blocking from different time-triggered slots does not overlap, the interference from all time-triggered classes J on stream i can be found by summation [8].

$$I_i^{CTB}(\Delta t) = \sum_{J \in \mathcal{E}} \left\lceil \frac{\Delta t}{t_J^{CYC}} \right\rceil \tilde{t}^{TT}_{cl(i),J} \tag{22}$$

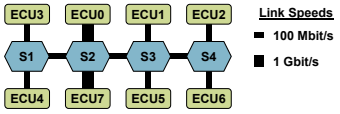


Fig. 5. Quad star topology

Preemption overhead: As the less-critical traffic streams cannot preempt themselves, there can only be one frame preemption per time-triggered slot. In any time interval Δt, the maximum number of preemptions can be found by summing over the maximum number of time-triggered slots of all critical (time-triggered) traffic classes J in Δt (cf. Eq. (22)). Multiplying this number with the preemption overhead divided by r_{TX} yields the preemption overhead (cf. Theorem 2).

$$I^{PO}(\Delta t) = \frac{24\,\text{bytes}}{r_{TX}} \sum_{J \in \mathcal{E}} \left\lceil \frac{\Delta t}{t_J^{CYC}} \right\rceil \tag{23}$$

Now, for any non-time-triggered traffic stream i, the worst-case queueing delay $w_i^P(q, a_i^q)$ of the q-th frame, which arrived at time a_i^q, under frame preemption can be computed.

$$w_i^P(q,a_i^q) = I_i^{LPB,P} + I_i^{SPB,P}(q,a_i^q) + I_i^{HPB}(w_i^P(q,a_i^q)) + I_i^{CTB}(w_i^P(q,a_i^q)) + I^{PO}(w_i^P(q,a_i^q)) \tag{24}$$

Then, the maximum frame transmission latency of the q-th frame of traffic stream i under IEEE 802.1Qbv with preemption can be computed very similarly to that under IEEE 802.1Q (Eqs. (15) and (16)). As argued before, the worst-case timing guarantees of critical traffic in IEEE 802.1Qbv are not affected by frame preemption and can be computed as presented in [8].

VII. Evaluation

Now, we use our analysis to evaluate the timing impact of IEEE 802.3br frame preemption in combination with IEEE 802.1Q and IEEE 802.1Qbv under worst-case conditions. We investigate a heterogeneous setup with multiple traffic streams of different criticalities (and priorities), including latency-critical control traffic, high-bandwidth ADAS traffic with large Ethernet frames, and low-priority (best effort) infotainment traffic. In our experiments, we vary the mapping of Ethernet traffic classes (priorities) to IEEE 802.3br's express and preemptable MAC interfaces and quantify how each traffic class benefits or suffers for each mapping. We use worst-case end-to-end latency guarantees as the comparison metric.

For our evaluation, we use the topology in Figure 5. The network traffic has been provided by Daimler AG and is summarized in Table I. There are three different traffic classes: control data traffic (CDT), general control traffic (GCT), and camera traffic (CAM). CDT traffic is assumed to be latency-critical and has the highest Ethernet priority (priority 4 in our example). GCT traffic is latency-sensitive, but not as critical as CDT. It is mapped to priority 3. CAM traffic is high-bandwidth video traffic and has more relaxed latency constraints. It is mapped to priority 2. Table I also describes how traffic is transmitted. There are unicast, multicast (the notation $n(d)$ indicates that there are n multicast streams to d destinations), and broadcast transmissions. For each traffic class the table also gives the minimum, maximum, and average payloads and

DOI: pending

periods of its traffic streams. We assume that the payload is transmitted using UDP/IP. Thus, 28 bytes of protocol overhead must be added. Frames of CDT, GCT, and CAM traffic are injected with a *periodic with jitter* event model [2]. The jitter is set to the period, to model that an occasional burst of two frames might be injected into the network. Additionally, two non-real-time (NRT) traffic streams with maximum frame size (1542 bytes) are broadcasted in opposite directions as low-priority interference on priority 1, originating from ECU2 and ECU4. Both streams have a period of 10 ms.

Figure 6 shows our evaluation results. Even though this is not a random experiment, we use boxplots to summarize the worst-case end-to-end latency guarantees of all streams of a given traffic class. For each traffic class, the box covers 50 % of the worst-case end-to-end latency guarantees, with its lower and upper borders giving the 25 % and 75 % quartiles, respectively. The whiskers indicate the worst-case guarantees of the streams with the shortest and longest latency guarantees. The median and average among the worst-case latency guarantees are marked by a red bar and a black dot, respectively. The x-axis labels identify our experiments, which we discuss next.

Standard Ethernet: We conduct four experiments: (*802.1Q*) IEEE 802.1Q without frame preemption as a baseline for comparison. (*e4p321*) Only CDT is mapped to the express MAC interface and GCT, CAM, and NRT are preemptable. (*e43p21*) CDT and GCT are mapped to the express MAC interface and CAM and NRT are preemptable. (*e432p1*) CDT, GCT, and CAM are mapped to the express MAC and only NRT is preemptable. Note that the experiment names indicate which traffic classes (priorities) are mapped to which MAC.

Figure 6a shows the results of these experiments. The boxplots are divided into three groups. The first group shows the results for CDT traffic under different mappings, while the second and third groups show the results for GCT and CAM traffic, respectively. By definition, the end-to-end latencies of NRT traffic do not require timing verification and are omitted.

For *e4p321*, the worst-case end-to-end latency guarantees of CDT improve significantly over *802.1Q* (60 % on average). As expected, the worst-case latency guarantees of GCT and CAM are worse than under *802.1Q*, due to the additional preemption overhead. Their latency degradation, however, is comparatively small. On average GCT anc CAM traffic have latency guarantees that are only 6 % and 2 % larger than under *802.1Q*, respectively.

In *e43p21*, the improvement of CDT is smaller than in *e4p321*, due to increased lower-priority blocking by GCT traffic, which cannot be preempted anymore. However, as GCT frames are comparatively small, CDT's average improvement is still 56 %. GCT, expectedly, improves and, compared to *802.1Q*, its worst-case latency guarantees are, on average, 33 % smaller. CAM traffic suffers from the additional preemption overhead from GCT and its latency guarantees are, on average, 7 % worse than in *802.1Q*.

TABLE I
QUAD STAR 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

In *e432p1*, the average improvements of CDT and GCT over *802.1Q* are 34 % and 25 % (respectively) as now they experience lower-priority blocking from large CAM frames, which are now also mapped to the express MAC. The average improvement of CAM traffic over *802.1Q* is 14 %.

Our experiments show that latency-critical traffic clearly benefits from frame preemption. In our setup, *e43p21* appears to be a good compromise, as the worst-case end-to-end latency guarantees of both control traffic classes improve significantly, while CAM traffic only experiences minor degradation.

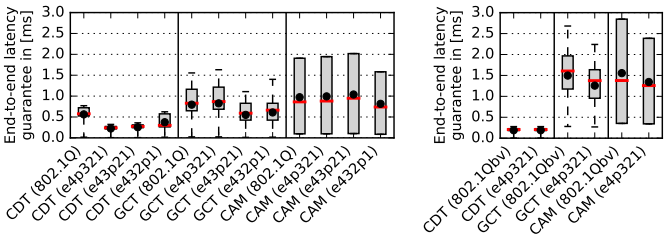
Ethernet TSN: For IEEE 802.1Qbv, we assume that CDT traffic is always mapped to a time-triggered slot, which is repeated with a period of 5 ms. We further assume that the time-triggered slots at each switch port are scheduled such that CDT traffic does not need to wait for its slot and that all slots are large enough to process any CDT traffic arriving within the slot. This was an important observation from [8], as otherwise IEEE 802.1Qbv gives in very poor worst-case latency guarantees. A slot length of 250 us is sufficient in our setup. We conduct two experiments: (*802.1Qbv*) IEEE 802.1Qbv without frame preemption as a baseline for comparison. (*e4p321*) Only CDT traffic is mapped to the express MAC interface and GCT, CAM, and NRT are preemptable. Mapping GCT to time-triggered slots would lead to very poor link utilization, due to the large variance of its periods (cf. Table I).

Figure 6b shows the results of these experiments. Again, the boxplots are divided into three groups showing the results for CDT, GCT, and CAM traffic. We compare *e4p321* to *802.1Qbv*. As expected, time-triggered CDT traffic does not benefit from the preemption of GCT and CAM, as (plain) IEEE 802.1Qbv already ensures that time-triggered traffic is scheduled at predefined times. This is implemented via guard bands, that block other traffic early enough (cf. Section VI).

GCT and CAM traffic, however, benefits from frame preemption, even though they are the preemptable classes. As GCT and CAM both use the preemptable MAC interface they can only be preempted by express traffic, but not preempt themselves (Section VI). Frame preemption, however, reduces the interference from the time-triggered CDT slots by reducing the guard band (see Eq. (21)). In our setup, the guard band is reduced by 90 %. The preemption overhead is comparatively small, as there can only be one preemption per time-triggered slot, i.e. every 5 ms. More precisely, the worst-case end-to-end latency guarantees of GCT and CAM improve by 15 % and 9 % (on average) compared to *802.1Qbv*, respectively.

Discussion: Comparing the results for IEEE 802.1Q and IEEE 802.1Qbv yields an interesting finding: as long as CAM traffic is not using the express MAC interface (i.e. stays preemptable), the worst-case end-to-end latency guarantees of CDT traffic are almost identical for IEEE 802.1Q and IEEE 802.1Qbv (*e4p321* and *e43p21* in Figure 6a vs. *802.1Qbv*

DOI: pending



(a) Standard Ethernet (IEEE 802.1Q) (b) TSN (IEEE 802.1Qbv)

Fig. 6. Worst-case end-to-end latency guarantees of the quad star topology

in Figure 6b). For *e4p321*, their maximum difference is 46 us and for *e43p21* it is 83 us (i.e. between 1 % and 1.7 % of the CDT period), which is very low compared to typical automotive latency requirements. This is because frame preemption in IEEE 802.1Q reduces the lower-priority blocking for express traffic to the technical minimum (Eq. (3)). This improvement vanishes, if long frames, e.g. from CAM traffic in *e432p1*, are introduced to the express MAC interface. Furthermore, the worst-case latency guarantees of GCT and CAM traffic under IEEE 802.1Qbv are worse than under IEEE 802.1Q.

Without frame preemption, similar (low) lower-priority blocking times could be achieved by permanently limiting the length of lower-priority frames. This, however, would entail a large overhead as more Ethernet frames are required and each additional frame requires 42 bytes of protocol overhead (each frame must include preamble, SFD, DA, SA, Q-Tag, ET, FCS, and IFG, see Figure 2). Frame preemption, in contrast, only splits frames into shorter fragments when required. As the periods of control traffic (CDT and GCT) are typically larger than those of CAM traffic (see Table I), not all CAM frames will be preempted. This can be observed in Figure 6a, where the average CDT and GCT improvement for experiments *e4p321* and *e43p21* is larger than the average CAM degradation. E.g. for *e43p21*, the latency guarantees of CDT and GCT traffic are reduced (improved) by 56 % and 33 % (respectively), while the latency guarantees of CAM traffic only increase (worsen) by 7 %.

Consequently, with IEEE 802.3br frame preemption, standard Ethernet can achieve comparable worst-case performance for control traffic as Ethernet TSN. This is a very useful result, as, in comparison to IEEE 802.1Qbv, IEEE 802.1Q is less complex to setup. It only requires the mapping of traffic streams to priorities, whereas IEEE 802.1Qbv additionally requires the computation of schedules for the time-triggered slots at each switch (port). Additionally, IEEE 802.1Qbv requires tight network-wide time synchronization to enforce its schedules.

VIII. Conclusion

IEEE 802.3br introduces frame preemption to Ethernet. It is designed to integrate seamlessly with other Ethernet standards, such as standard Ethernet (IEEE 802.1Q) and Ethernet TSN. IEEE 802.3br only allows one level of preemption by specifying an express (non-preemptable) and a preemptable MAC interface. In this paper we presented a formal timing analysis for frame preemption under IEEE 802.3br. We evaluated the effect of preemption on the worst-case end-to-end latency guarantees of standard Ethernet and Ethernet TSN. Our experiments show that, in a typical automotive setup, the latency guarantees of non-preemptable express traffic show large improvements, while, at the same time, preemptable traffic does not degrade much. Interestingly, with frame preemption, standard Ethernet's latency guarantees for express traffic are very close to those of Ethernet TSN, due to reduced blocking from preemptable lower-priority traffic. This makes standard Ethernet an interesting alternative to TSN even for latency-critical traffic, as it is less complex to setup, e.g. no forwarding schedules and no tight time synchronization are required.

References

[1] IEEE Time-Sensitive Networking Task Group, "IEEE 802.1Qbv - Enhancements for Scheduled Traffic," https://standards.ieee.org/findstds/standard/802.1Qbv-2015.html.

[2] 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.

[3] 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.

[4] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying Trajectory Approach with Static Priority Queuing for Improving the Use of Available AFDX Resources," *Real-Time Systems*, vol. 48, no. 1, Jan. 2012.

[5] K. Revsbech, H. Schiøler, T. K. Madsen, and J. J. Nielsen, "Worst-case Traversal Time Modelling of Ethernet Based In-Car Networks Using Real Time Calculus," in *Proceedings of NEW2AN*. Springer-Verlag, 2011.

[6] 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.

[7] 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.

[8] 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.

[9] W.-K. Jia, G.-H. Liu, and Y.-C. Chen, "Performance Evaluation of IEEE 802.1Qbu: Experimental and Simulation Results," in *IEEE Conference on Local Computer Networks (LCN)*, 2013.

[10] J. Kim, B. Y. Lee, and J. Park, "Preemptive switched ethernet for real-time process control system," in *IEEE Conference on Industrial Informatics (INDIN)*, 2013.

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

[12] V. Mikolasek, A. Ademaj, and S. Racek, "Segmentation of Standard Ethernet Messages in the Time-Triggered Ethernet," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2008.

[13] IEEE P802.3br Interspersing Express Traffic Task Force, "P802.3br - Standard for Ethernet Amendment Specification and Management Parameters for Interspersing Express Traffic," https://standards.ieee.org/develop/project/802.3br.html.

[14] IEEE Time-Sensitive Networking Task Group, "802.1Qbu - Frame Preemption," http://www.ieee802.org/1/pages/802.1bu.html.

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

[17] 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.

[18] L. Winkel and M. J. Teener, "Real-time Ethernet on IEEE 802.3 Networks," in *IEEE 802.3 Plenary Meeting*, Berlin, Germany, 2015.

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