Improving Worst-case TSN Communication Times of Large Sensor Data Samples by Exploiting Synchronization

JONAS PEECK and ROLF ERNST, TU Braunschweig, Germany

Higher levels of automated driving also require a more sophisticated environmental perception. Therefore, an increasing number of sensors transmit their data samples as frame bursts to other applications for further processing. As a vehicle has to react to its environment in time, such data is subject to safety-critical latency constraints. To keep up with the resulting data rates, there is an ongoing transition to a Time-Sensitive Networking (TSN)-based communication backbone. However, the use of TSN-related industry standards does not match the automotive requirements of large timely sensor data transmission, nor it offers benefits on time-critical transmissions of single control data packets. By using the full data rate of prioritized IEEE 802.1Q Ethernet, giving time guarantees on large data samples is possible, but with strongly degraded results due to data collision. Resolving such collisions with time-aware shaping comes with significant overhead. Hence, rather than optimizing the parameters of the existing protocol, we propose a system design that synchronizes the transmission times of sensor data samples. This limits network protocol complexity and hardware requirements by avoiding tight time synchronization and time-aware shaping. We demonstrate that individual sensor data samples are transmitted without significant interference, exclusively at full Ethernet data rate. We provide a synchronous event model together with a straightforward response time analysis for synchronous multi-frame sample transmissions. The results show that worst-case latencies of such sample communication, in contrast to non-synchronized approaches, are close to their theoretical minimum as well as to simulative results while keeping the overall network utilization high.

CCS Concepts: • **Computer systems organization → Real-time systems**; *Embedded and cyber-physical systems*; • **Networks → Packet-switching networks;**

Additional Key Words and Phrases: Safety, real-time, Ethernet, verification, automated driving, TSN

ACM Reference format:
Jonas Peeck and Rolf Ernst. 2023. Improving Worst-case TSN Communication Times of Large Sensor Data Samples by Exploiting Synchronization. *ACM Trans. Embedd. Comput. Syst.* 22, 5s, Article 139 (September 2023), 25 pages.
https://doi.org/10.1145/3609120

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Embedded Software (EMSOFT), 2023.
This research is accomplished within the project "AUTOtech.agil" (FKZ 01IS22088A). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).
Authors' address: J. Peeck and R. Ernst, TU Braunschweig, Braunschweig, Germany; emails: {peeck, ernst}@ida.ing.tu-bs.de.

1539-9087/2023/09-ART139 $15.00
https://doi.org/10.1145/3609120

1 INTRODUCTION

On the way to higher degrees of driving automation, data is collected, passed, and processed along the so-called sensor to actuator cause-effect chains (CECs), that span over multiple Electronic Control Units (ECUs). Here, data types can be e.g., sensor, control, but also infotainment, or diagnosis data. Sensor data for Advanced Driver-Assistance Systems (ADAS) applications, in particular, is no longer processed exclusively in local domains – e.g., to feed an anti-lock braking system –, but is communicated throughout the vehicle for further preprocessing or sensor data fusion in terms of highly automated driving. Therefore, traditional local bus systems, such as CAN or FlexRay, are not suitable candidates to be used as a communication backbone, as their data rate is way too low. As a consequence, the automotive industry shifted towards a *Zonal Architecture* [8, 35], where zone gateways are connected to an Ethernet backbone, as a preferred zonal topology [26]. Sensor data transmission can access the backbone in order to be transmitted to another local zone for further processing. Even though the Ethernet technology is well-approved, the consumer Ethernet following IEEE 802.3 does not have serious safety requirements for data transfer and, hence, lack mechanisms for timing guarantees. Without further effort, it is not possible to comply with automotive technical safety requirements, such as prescribed in the ISO 26262 on functional safety [18]. From this it follows that the interference between **data streams** must be limited in order to comply with maximum technical safety requirements, which are modeled by end-to-end latency guarantees of CECs.

Periodic data acquisition in sensors is usually triggered synchronously. Thereby, along the CEC, data is commonly passed between software components following sporadic communication:

Definition 1.1. **Sporadic communication** directly transmits data if generated by software tasks. This propagates the jitter of such tasks along the CEC. Hence, in terms of our work, sporadic refers to an *unsynchronized periodic communication with jitter*.

Due to the synchronized start of an automated driving CEC as well as the non-synchronized and jittered processing of the tasks along the CEC, simultaneous communication times of data samples over the network, which lead to data collisions, can not be avoided. Thus, by increasing the number of sensor data communication to the network, congestion effects can accumulate even more and let bounds on worst-case sample communication times become unacceptably high. Overall, such a design approach is not capable to provide low latency guarantees for network communication, as required in safety-critical highly automated vehicles.

Figure 1 depicts such an example, where the paths of two sensor data streams include the same link from *Switch 1* to *Switch 2*. Each sporadic transmission of a sensor data stream is represented by a large **sensor data sample**, that has to be split up into multiple Ethernet frames for transmission. If both frame bursts arrive at the shared output port of *Switch 1* towards *Switch 2* simultaneously, they interfere, queue up, and have to be arbitrated. In detail, as a receiving application needs a complete sensor data sample – like an image – to start processing, the delay of the last frame of a frame burst determines the *sample latency*. Hence, such uncontrolled queuing causes significant transmission delay in a CEC, which might not be tolerable in terms of its end-to-end latency requirement. Unfortunately, while the shared link is over-utilized, the subsequent non-shared links leaving *Switch 2* are idle half the time.

To overcome the described transmission interference in order to comply with safety standards, a current approach is to add complexity to the network switches and Network Interface Cards (NICs). Here, different standards are gathered under the term TSN. An important option of TSN is IEEE 802.1Qbv, which specifies the *Time Aware Shaper (TAS)* [1]. The TAS implements a Time Division Multiple Access (TDMA)-based global transmission schedule, that relies on time synchronization of all network hardware components. In detail, software components can pass data to a

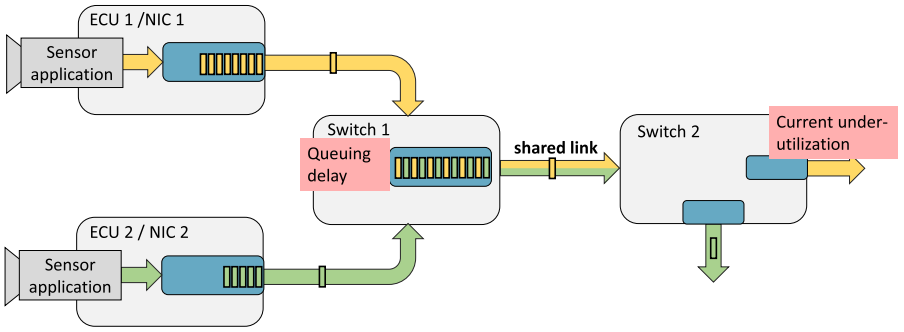


Fig. 1. Two sensor data streams transmit their multi-frame sensor data samples over a switched Ethernet network. After being passed to their NICs for transmission, the frames access the same outgoing link from Switch 1 to Switch 2 at the same time and queue up for arbitration in the corresponding output port.

TSN NIC sporadically at any time and the network infrastructure should handle the transmission based on synchronized TDMA transmission slots. Thereby, the TSN TDMA schedule is **designed for single frame** transmissions of small control data in industrial setups [7]. Tight time synchronization between NICs and switches enables a precise shift along hops in their TDMA schedules, which allows single frames to be fast-forwarded without delay via the multi-hop TSN network. However, such a design model does not match the transmission of large **multi-frame sensor data samples** as described above. It rather delays the sample transmission in case only one frame transmission in a TDMA cycle is assigned to a certain ADAS data stream. In this context, TSN is not work conserving, so that unused control data TDMA slots cannot be assigned to ADAS streams. Hence, to transmit a sample fully, TSN would need a large amount of TDMA cycles, which makes the fast forward of individual frames obsolete. In contrast, widening the size of a TSN cycle to be able to include a full sensor data sample would block a small single frame control data message for several milliseconds, which is unacceptable under given automotive requirements. Altogether makes TSN less efficient and unnecessarily complicated to use and configure in case that large time-critical ADAS data should be transmitted along with other time-critical small control data. In the avionic domain following ARINC 653 [27], TDMA-based time partitioning concepts with an overall privileged partition are described for software execution on ECUs. However, TSN does not provide the possibility of a privileged task that has scheduling priority over the TDMA schedule, e.g., for latency-critical control data traffic.

In this work, to provide temporal isolation between timely sample data transmission, we propose another design approach, which avoids data rate reductions for single sensor data sample transmissions and still keeps the latency of critical control frames low. First, we keep the network hardware as simple as possible, in order to minimize the configuration complexity of the hardware and to provide high transmission predictability. Therefore, we stick to IEEE 802.1Q Ethernet [3]. In the context of our work, it is only important that it allows packet prioritization and follows a strict First In-First Out (FIFO) schedule. Formally, 8 priority classes are supported, however, actual hardware implements 4, which are assigned to automotive traffic classes as follows [34]:

(1) **Control data**, e.g., steering or power train: Small messages, which are latency-critical in terms of safety.
(2) **ADAS (sensor data) streams:** Large data samples, which are latency-critical in terms of safety.

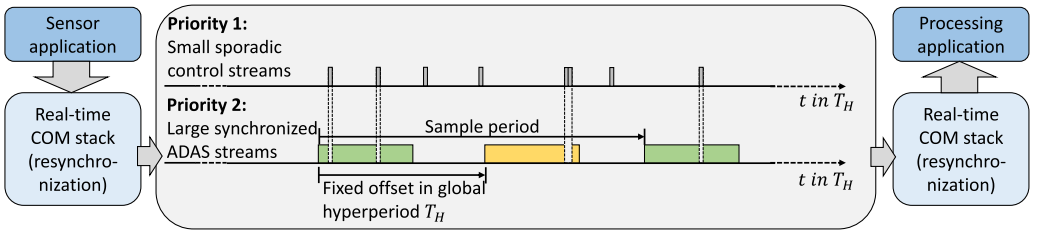


Fig. 2. Conceptual view of two synchronized ADAS sensor data sample transmissions. Different offsets place them to non-overlapping times in the globally synchronized hyperperiod T_H. Interference is only subject to a small number of control data frames.

(3) **Infotainment:** Large data streams, which are not safety-critical. They can be pre-buffered in terms of latency.
(4) **Other best-effort communication:** File transfer, diagnostics, ...

To ensure time guarantees of safety-critical small control data we let such messages to be transferred without restriction. Based on the chosen prioritization, this allows very low latencies as there is no interference with other traffic classes. Moreover, control data is too small to cause noteworthy blocking effects on lower-priority large ADAS sensor data transmissions.

After simplifying the network hardware as well as taking care of low latency control data transmission, the next part of our novel design approach is to synchronize ADAS data communication times. As depicted in Figure 2, sensor data transmissions (priority 2) are located to dedicated times within a global **hyperperiod** T_H. Here, different offsets related to that hyperperiod lead to fixed shifts in transmission times between the samples corresponding to the green and yellow ADAS streams. The simple but very effective idea is that if samples are sent at different times, interference leading to critical latency growth (cf. Figure 1) can not occur. As a precondition, we need time synchronization in the ECUs, which is commonly provided via the Precision Time Protocol (PTP) as standardized in IEEE 1588 [4]. Based on this, we define synchronous communication in contrast to sporadic communication as follows:

Definition 1.2. **Synchronous communication** holds back the data transmission corresponding to an offset in order to *resynchronize* at specific times in the hyperperiod. The remaining jitter is reduced to the one of the COM stack and therefore removes the one of the predecessing software components of the CEC.

Representing the synchronization accuracy, the jitter in our work is allowed to be significantly larger than the required accuracy of TSN synchronization of TAS-related TDMA cycles, which makes system configuration and parameterization less complex. Based on the remaining COM stack jitter, different offset times can directly be the result of different execution times of e.g., preprocessing tasks, as well as the flexibility of task to ECU mappings in a CEC. Such offsets then remove interference by synchronizing transmission times and therefore significantly improve latencies of a large number of simultaneously active ADAS streams. Avoiding TSN TDMA slots in that context lets the network remain load conserving so that the entire data rate can be maintained.

Lower-priority data, such as infotainment, is related to **soft deadline requirements**. In case it should have transmission opportunities within ADAS frame bursts, a COM stack can implement slight frame burst shaping such as provided by audio/video bridging (AVB), but based on software. Moreover, also the slack between two ADAS transmissions can be utilized. While latencies of safety-critical applications are verified by a formal analysis (as done in this work), in industry, it is sufficient to evaluate lower-priority traffic by simulation. Note that including lower-priority traffic is not part of this work, as we focus on the basic synchronization effects of timely ADAS data.

Contribution: This work introduces and evaluates a novel communication approach that synchronizes ADAS data sample transmissions in a global hyperperiod only based on assigned offsets. Therefore, we formalize the synchronous offset-based transmission model along given network paths. As the single frame approach of TSN is unable to efficiently transmit larger ADAS streams, this synchronized communication approach proves to be highly efficient in order to provide low latency guarantees for such multi-frame transmissions. To prove this claim, we provide an evaluation based on a straightforward response time analysis and simulation. As complexity is not within routing, especially not in current automotive network topologies, we show how even a non-optimized configuration can significantly outperform existing sporadic approaches including TSN. Our work is thus the basis for a fundamental change in the design of automotive Ethernet communication that enables the efficient integration of low-latency multi-frame sensor data transmission.

The remainder of this work is structured as follows. In Section 2, we discuss software techniques on ECUs that are capable of implementing the proposed synchronous design approach, as well as work on the response time analysis to determine upper analytical bounds for synchronous sample transmissions. We specify the synchronous system model in Section 3. Based on that model, Section 4 introduces straightforward offset and route configuration methods for ADAS streams. To prove the effectiveness of our synchronous design approach, we provide a synchronous response time analysis in Section 5. Then, Section 6 evaluates synchronous configurations by simulation and analysis. In addition, we compare them to non-synchronized setups by analysis as well as simulation and provide a detailed comparison to TAS. Last, we conclude in Section 7.

2 RELATED WORK

As argued in Section 1, state-of-the-art hardware solutions, such as TSN-related standards, but also the sporadic communication approach in general, do not meet the demands of large data rate communication that follow highly automated driving. To use synchronization of sensor data sample transmissions at the network input, a sufficiently accurate timed programming technique is required. A common approach is the Logical Execution Time (LET) programming paradigm, which delays communication of one task to another until a predefined upper logical execution time bound [19]. LET was commonly considered to be based on TDMA [14]. However, it has become established that LET can also be used under Static Priority Preemptive (SPP) scheduling to preserve load processing capabilities. The most recent development of LET is the extension towards System Level-LET (SL-LET), which includes the network communication in a CEC and interprets transmission of data samples as separate *interconnect tasks* [13]. This technique is deployed in automotive systems [15]. While the logical delay of SL-LET allows the required resynchronization in a hyperperiod, the synchronization uncertainty decreases to the COM stack jitter, which can be up to 4 ms [16]. Here, the recent development of a real-time capable COM stack could demonstrate how such jitter can be further reduced to below 500 µs by low-level packet prioritization [16].

Following the given mechanisms of resynchronization along data sample transmission, the problem remains how to determine worst-case response times of synchronous sample communication efficiently. For TSN related AVB substandards, [12] could show that shaping of traffic classes, even if they might benefit from the shaping of higher priority load, always leads to higher worst-case response times. Interestingly IEEE 802.1Q has led to the best results, hence, its selection in our synchronous design approach seems appropriate. In that context [31] exploited the analyzed IEEE 802.1Q worst-case response times under FIFO scheduling while assuming sporadic communication. Here, the idea is that if once an Ethernet frame is ahead of another frame, it can no longer be disturbed by it. This way, the analytical pessimism of single-frame transmission could be further reduced. In addition, [30] introduced the *multiple event busy time* to tighten the jitter propagation

in the event-propagation step of the Compositional Performance Analysis (CPA). This method has become the standard event propagation model in CPA. It is particularly useful for larger task chains as shown for a multiprocessor system [30] or an MpSoC considering memory accesses [29], but also given for multi-hop Ethernet communication. Common to all of the presented analysis approaches is that they assume the sporadic communication model for frame transmissions (cf. Definition 1.1). For that model, under worst-case assumptions, the analysis always assumes the *critical instant*, which considers the maximum event arrival of all load sources at the same time. In the context of ADAS communication, it is then always assumed that two sensor data sample transmissions traveling the same link can collide. In contrast, resynchronization is designed to avoid such a case, hence, the sporadic design approach is clearly not usable for our work. As we will show, it will lead to intolerable high latencies in analysis as well as in simulation.

Similar to event model-based analysis techniques such as CPA, also approaches that are based on service curves, such as Network Calculus [20], are able to analyze worst-case response times of streams. However, they lack the same limitations as they are considering a sporadic model in which the amount of interference between transmissions over the same link cannot be related to their placement in a hyperperiod. The burst pre-shaping approach of [21] that targets ADAS Ethernet communication is therefore limited by that interference within the deadline of each ADAS sample.

Towards offset-based synchronized task activation, that could also be used to evaluate the performance of synchronized IEEE 802.1Q communication, early approaches focused on single tasks with a fixed jittered offset [33]. Based on this, further work extended the model towards dynamically changing offsets, thus allowing for temporary task suspension in multi-core systems [23]. This way, offsets are allowed to be even larger than the period. Optimizations on pessimism and analysis performance in the context of a hyperperiod have been presented in [28]. Other work has used appropriate offset assignments to increase local system properties, like the response times of communication over an automotive CAN Bus [9, 17]. In the avionic systems domain, [24] adapted the offset-based response time analysis towards hierarchically-scheduled time-partitioned distributed systems. All these presented approaches have in common, that they only model task activation individually. Thus, they are not capable of modeling multi-frame communication along switched Ethernet components, which are representatives of task chains.

Overall, already used design approaches such as SL-LET provide resynchronization, however, its effectiveness in system design as well as in analysis has not been addressed yet. Consequently, the possible latency gains are not exploited yet. Therefore, in the next section, we begin with the formulation of a suitable synchronous design before demonstrating its benefits through simulation and a proposed synchronous analysis.

3 SYSTEM MODEL

In this section, we first specify the system model elements as well as their mapping in Section 3.1. Therefore, all relevant parameters are gathered in Table 1. It follows the description of frame transmission event models including their propagation along Ethernet streams in Sections 3.2 and 3.3 as well as the principles of Static Priority non-Preemptive (SPNP) scheduling in Section 3.4. As a result, we obtain a full system model together with its transmission and scheduling principles that are the base for the later response time analysis and simulation.

3.1 Communication Resources and Task Chains

In the Ethernet network used in this work and depicted in Figure 3, communication capacities are provided by switches and NICs. Specifically, on each side of a bidirectional wired connection between two network components, an **output port** represents a **communication resource** r to transmit frames [11]. A frame transmission on an output port is modeled as a **communication**

Table 1. Basic Parameters of the Synchronous ADAS Communication Model

Parameter	Description
r	Communication output port resource
$\tau_{i,c}$	Communication task i of task chain c
S_c	Sequence of tasks representing a task chain c
S_i^+	Maximum frame size of the i-th communication task
R_r	Data rate of an output port resource r
C_i^+	WCCT of the i-th communication task
T_H	The global hyperperiod
T_c	Sample period of a task chain c
N_c	Sample-related frame burst size of task chain c
d_c	Frame period of task chain c
J_c	Jitter of task chain c
φ_c	Task chain offset of synchronous periodic sample transmission
$\delta_{\text{sy},i/c}^{-/+}(n)$	Lower/upper synchronous event arrival bound functions on task/chain level
$\eta_{\text{sy},i/c}^{-/+}(t)$	Lower/upper synchronous event arrival time function on task/chain level
$R_i^+(n)$	Worst-case response time of the n-th communication task activation.

task $\tau_{i,c}$ that is mapped to the output port resource r. Here, i is the task's unique ID and c denotes its affiliation to a certain task chain. Thereby, a **communication task chain** S_c is a sequence of communication tasks $S_c = \{\tau_{i,c}, \tau_{i+1,c}, ...\}$ each mapped to subsequent output ports of directly connected Ethernet NICs or switches. The resource allocations of such a task chain represent a frame transmission through the Ethernet network, whereby the termination of a task's communication time represents an **activation event** of its subsequent task in the chain. In Figure 3 a frame transmission from NIC 7 to NIC 2 is described by the chain $S_1 = \{\tau_{1,1}, \tau_{2,1}, \tau_{3,1}, \tau_{4,1}\}$. In this work, under worst-case considerations, we are only interested in the Worst-Case Communication Times (WCCTs) of frames assuming maximum frame sizes S_i^+, corresponding to the transmitting task. Then the WCCT of a task, denoted as C_i^+, depends on that size, as well as the resource's data rate R_r as follows:

$$C_i^+ = \frac{S_i^+}{R_r} \tag{1}$$

Note that in a task chain sequence, all tasks have the same maximum frame size S_i^+ as they represent the transmission of the same frame.

A frame transmission along a task chain is modeled as an external activation of its first task from a **chain event source**. In state-of-the-art timing models, such a source is described by a number of n activations that can occur within a lower and upper time interval Δt. This event number to time interval mapping is denoted as the minimum and maximum distance functions $\delta_c^-(n), \delta_c^+(n)$. Thereby, c indicates that the event source function is related to the first task $\tau_{i,c}$ in the chain c. This work considers both, sporadic and synchronous transmission of data, e.g., corresponding to control or synchronous ADAS streams. Hence, we distinguish two different notations on that functions, which are $\delta_{\text{sy},c}^-(n), \delta_{\text{sy},c}^+(n)$ for synchronous task chains and $\delta_{\text{sp},c}^-(n), \delta_{\text{sp},c}^+(n)$ for sporadic ones. In terms of large sample data transmissions, such minimum distance functions describe periodic frame bursts. In detail, a sample is transmitted with the **sample period** T_c and consists of a **number of frames** N_c for which each frame corresponds to an activation of the first task in the chain. Frame activation in a task chain is subject to a **frame period** d_c as well as a **jitter** J_c. In the following, we will specify event source functions based on that parameters related to both sporadic and synchronous task chains.

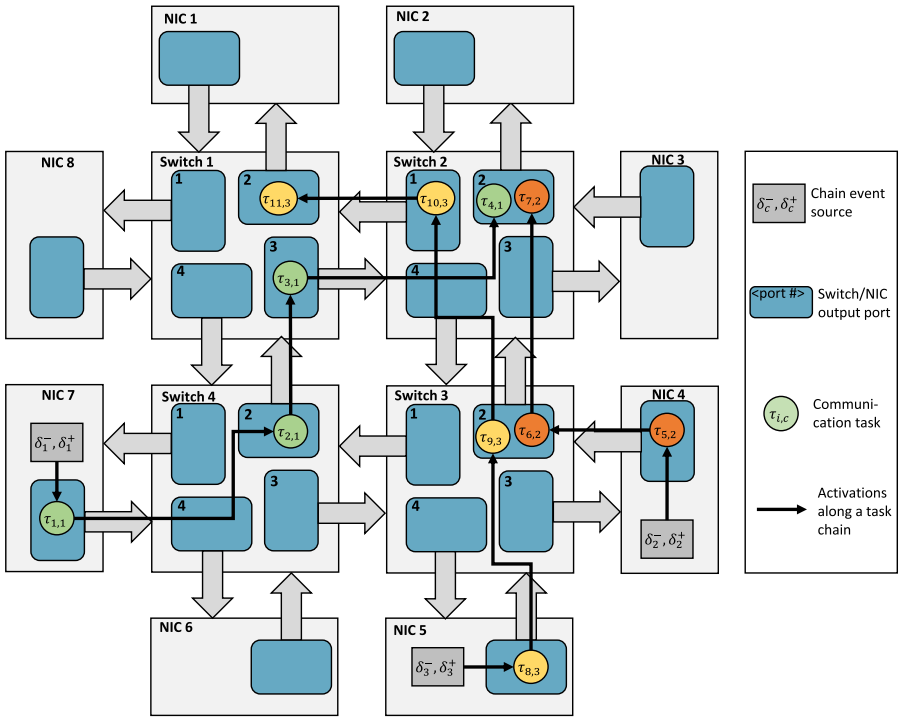


Fig. 3. Communication task chain to resource mapping for a widely deployed automotive Ethernet ring topology. Each task representing a frame transmission is mapped to a NIC or switch output port and is activated by either an event source or its predecessor in the chain. The numbers in the output ports denote their port ID.

3.2 Sporadic Event Model

On the highest scheduling priority, control data is transmitted based on a non-synchronous sporadic approach. This data is only up to one frame. Modeling sporadic ADAS data communication, which is used as a comparison to our synchronous ADAS approach, requires multiple frame transmissions per period. Hence, to match both sporadic control and sporadic ADAS streams, each task chain has a **sample period** T_c, a per-sample **number of frames** $N_c \geq 1$, a frame distance d_c, as well as a transmission jitter J_c. The resulting *(sporadic) minimum distance function* $\delta^-_{\mathrm{sp},c}(n)$, which is relevant for sporadic worst-case response time analysis, is defined as follows:

$$
\begin{aligned}
0 \leq n < 2: \quad & \delta^-_{\mathrm{sp},c}(n) = 0 \\
2 \leq n: \quad & \delta^-_{\mathrm{sp},c}(n) = \max\left\{0, \underbrace{\left\lfloor \frac{n-1}{N_c} \right\rfloor \cdot T_c}_{\text{Sample periodicity}} + \underbrace{((n-1) \mod N_c) \cdot d_c}_{\text{Frame periodicity}} - J_c \right\}
\end{aligned} \tag{2}
$$

Figure 4 depicts $\delta^-_{\mathrm{sp},c}(n)$ for an exemplary periodic transmission of 2 frames per data sample. Here, frame bursts of size $N_c = 2$ occur each T_c. Note that larger ADAS data is just modeled with larger N_c values, control data with $N_c = 1$. However, $\delta^-_{\mathrm{sp},c}(n)$ only describes the minimum distance between two frame transmission events of the same communication task in a chain. This makes it unable to model synchronized event arrival relations between *different* tasks on that resource. To

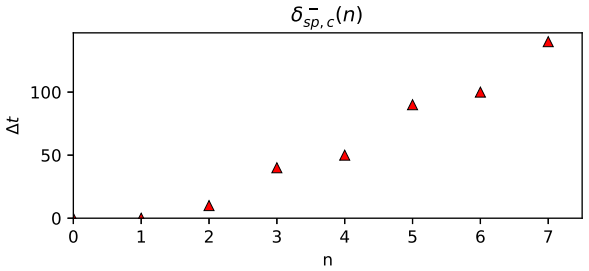


Fig. 4. Minimum distance function of a sample transmission with: $T_c = 50$, $N_c = 2$, and $d_c = 10$.

solve this, we will show in the following how to model event sources that represent offset-based synchronized placement of sample transmissions within a hyperperiod.

3.3 Synchronous Event Model and Event Propagation

As motivated in Section 1, synchronizing ADAS sample data communication is done by assigning offsets to transmissions in order to place them into a global hyperperiod T_H. Therefore, the location of each frame transmission in the hyperperiod is bounded by the tasks **lower and upper synchronous event arrival bound functions** $\delta^-_{\text{sy},c}(n)$ and $\delta^+_{\text{sy},c}(n)$. Considering an individual task chain, the principle of such synchronous functions along the task chain is depicted in Figure 5 and described in the following. Therefore, we first start with the event model from the chain event source, as it is well-controllable by the COM stack. We then continue with the event propagation along the chain.

Chain event source: A chain's event source representing the COM stack forcing a synchronous transmission is defined by the *lower and upper synchronous event arrival bound functions*:

$$\delta^-_{\text{sy},c}(n) = \underbrace{T_c \cdot \left\lfloor \frac{n}{N_c} \right\rfloor + \varphi_c}_{\text{Sample level activation}} + \underbrace{(n \mod N_c) \cdot d_c}_{\text{Frame level activation}} \quad , n \in \left\{0, 1, \ldots, \frac{T_H}{T_c} \cdot N_c\right\}, T_H \mod T_c \stackrel{!}{=} 0 \quad (3)$$

$$\delta^+_{\text{sy},c}(n) = \delta^-_{\text{sy},c}(n) + J_c \quad , n \in \left\{0, 1, \ldots, \frac{T_H}{T_c} \cdot N_c\right\}, T_H \mod T_c \stackrel{!}{=} 0 \quad (4)$$

Here, the synchronous event arrival bound functions of the first task $\tau_{i,c}$ in a chain **are identical** to the functions of the chain itself: $\delta^-_{\text{sy},i}(n) = \delta^-_{\text{sy},c}(n)$ and $\delta^+_{\text{sy},i}(n) = \delta^+_{\text{sy},c}(n)$. In Equation (3), T_c is the chain's **sample period**. To achieve a repeating pattern over all task chains each hyperperiod, T_c must be a divisor of T_H in the sense that the remainder is zero. A common value for T_H is 1 s. Next, to control the activation times on sample level in the hyperperiod, the **task chain activation offset** φ_c statically shifts each sample within T_H. Besides the control on sample level, an ADAS sample itself consists of multiple Ethernet frames. Hence, the offset shifts a burst of N_c task activations, where each burst has a distance of d_c between its frames. Last, as a piece of software, the COM stack adds up jitter J_c to the activation of frames in a task chain. Since a wake-up of the synchronized COM stack is based on an interrupt, the jitter can only add up to the offset instead of varying around the targeted activation time. Hence, for the lower synchronous event arrival bound function $\delta^-_{\text{sy},c}(n)$ (cf. Equation (3)) the jitter is zero, while the upper synchronous event arrival bound function $\delta^+_{\text{sy},c}(n)$ (cf. Equation (4)) adds up J_c.

The opposite relation from t in T_H to n is either the **lower or upper synchronous event arrival time function** $\eta^{-/+}_{\text{sy},i}(t)$. As depicted in Figure 5 for task $\tau_{i+1,c}$, it specifies the lower and upper number of event arrivals in an interval $[0, t]$ for a certain t related to T_H. We can determine

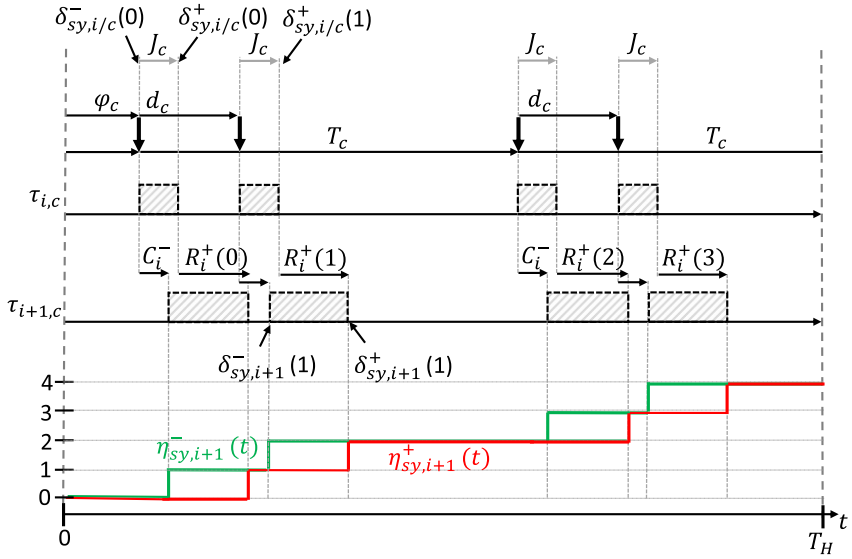


Fig. 5. Synchronous event (propagation) models of a task chain depicted in their global hyperperiod. Here, the sample period T_c is half the hyperperiod: $T_H = 2 \cdot T_c$.

$\eta^{-/+}_{\text{sy},i}(n)$ functions from their corresponding $\delta^{-/+}_{\text{sy},i}(n)$ functions as follows:

$$\eta^{-/+}_{sy,i}(t) = \underbrace{\left|\left\{n | \delta^{-/+}_{sy,i}(n) \mod T_H \leq (t \mod T_H), n \in \left\{0, 1, \ldots, \frac{T_H}{T_c} \cdot N_c\right\}\right\}\right|}_{\text{Within one hyperperiod}} + \underbrace{\frac{T_H}{T_c} \cdot N_c \cdot \left\lfloor \frac{t}{T_H} \right\rfloor}_{\text{Per hyperperiod}} \quad (5)$$

In the example, the time in the hyperperiod up to t is evaluated, and the earliest/latest event arrivals described by $\delta^{-/+}_{\text{sy},i+1}(n)$ are summed up. As n subsequent event arrivals do not have to be located in the exact same hyperperiod, we can exploit the repeating behavior each T_H by assigning the modulo operation to $\delta^{-/+}_{\text{sy},i+1}(n)$. Second, based on the repetitive activation pattern each hyperperiod, t values larger than T_H add the full number of $n = \frac{T_H}{T_c} \cdot N_c$ events for each time T_H fully fits into t: $\lfloor \frac{t}{T_H} \rfloor$.

Event propagation: Following a task chain sequence S_c, finishing the execution of a task $\tau_{i,c}$ represents an activation of its successor $\tau_{i+1,c}$ along the chain, because the frame is now waiting for transmission in the next output port (cf. Figure 5). Hence, event models are propagated from $\tau_{i,c}$ towards $\tau_{i+1,c}$. In the context of a formal response time analysis, the lowest possible activation bound of the n-th frame represented by $\delta^-_{\text{sy},i+1}(n)$ is given by adding the Best-Case Communication Time (BCCT) C_i^- to the lower bound of its predecessor, which is $\delta^-_{\text{sy},i}(n)$. The upper n-th frame activation bound $\delta^+_{\text{sy},i+1}(n)$ is represented by $\delta^+_{\text{sy},i}(n)$ plus the frame's corresponding Worst-Case Response Time (WCRT) $R_i^+(n)$:

$$\begin{aligned} \delta^-_{\text{sy},i+1}(n) &= \delta^-_{\text{sy},i}(n) + C_i^- \\ \delta^+_{\text{sy},i+1}(n) &= \delta^+_{\text{sy},i}(n) + R_i^+(n) \end{aligned} \quad (6)$$

Here, $R^+_{\text{sy},i}(n)$ is always referring to the n-th event starting exactly at $\delta^+_{\text{sy},i}(n)$, because in a synchronous model this represents the latest possible time in T_H up to which the frame finishes. This also maximizes the jitter to the next task in the chain.

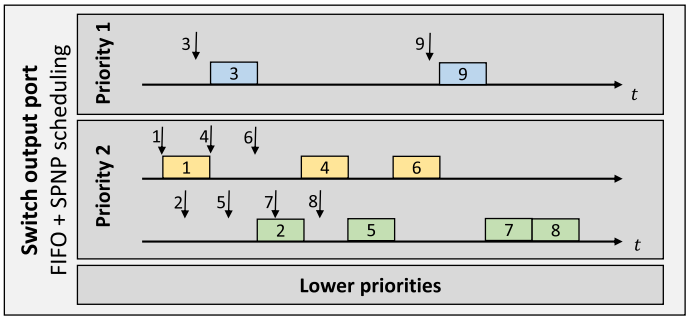


Fig. 6. SPNP and FIFO scheduling policy in pure IEEE 802.1Q Ethernet output port resources.

3.4 Static Priority Non-Preemptive Scheduling

Each output port represents access to an outgoing link from either a switch or a NIC. In case two or more communication tasks are mapped to such an output port (cf. $\tau_{4,1}$ and $\tau_{7,2}$ in Figure 3), their activation – representing frame transmissions – have to be arbitrated. Figure 6 shows an example of the SPNP with FIFO scheduling policiy as deployed by IEEE 802.1Q. For simplicity, events are named following their timely occurrence independent of their priority. Initially, the port is idle, hence, frame 1 of priority 2 can be directly transmitted after arrival. Due to the prioritization, frame 3 of priority 1 is transmitted next, even though it has arrived later than frame 2 of the lower priority. Following the non-preemptive scheduling, frame 3 has to wait at least for frame 1 to finish. Only looking at priority 2, frames are strictly transmitted in FIFO order following their order of arrival. The only interfering transmission is again frame 9 which has a higher priority.

4 SYNCHRONOUS ADAS STREAM CONFIGURATION

Following the synchronous event bound functions (cf. Equations (3) and (4)), ADAS data transmission can be placed to a certain time in the hyperperiod. To benefit from that synchronization, it is favorable to place different transmissions in a non-overlapping way. Depending on the current stream route allocations this might be not achievable for any transmission on any passed output port. Hence, we aim to minimize the resulting interference of one stream with others along its route. Figure 7 depicts the basic principles for offset configuration of *Stream 2* along its given output port route. Here, in presence of the already placed *Stream 1* we want to place *Stream 2* by the following steps:

(1) We increase the offset φ_2 incrementally by 1 ms, which shifts *Stream 2* in the hyperperiod.
(2) The propagation to next resources is assumed with no interference. This corresponds to an additional shift of the BCCT C_i^- towards the next output port resource.
(3) Separately on each resource, we evaluate the overlapping time of *Stream 2* for all of its individual sample transmissions to existing streams (here to *Stream 1* on *output port 1*). Therefore, we assume the undisturbed WCCT interval I_c of *Stream 2* and add two times a **sample robustness margin** M.
(4) To obtain the offset, we minimize the maximum cumulative overlapping of that interval to other already placed streams on any of the passed output ports and for any sample transmission of *Stream 2*.

Following these steps, some overlapping at higher utilization might occur, however, this only represents interference between a limited number of streams and not, as in the sporadic model, between all streams passing the same link. In general, we will receive a configuration in which most streams

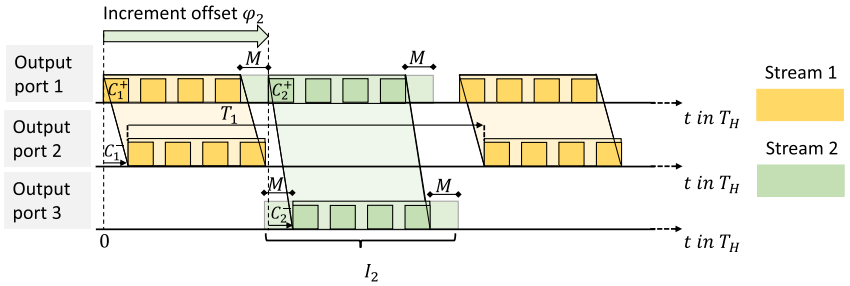


Fig. 7. Placement of a protected (green) ADAS stream in the hyperperiod under the existence of another already placed one. The described procedure results in a non-overlapping configuration including the *sample robustness margins*.

are placed directly one after another at a distance of M. This margin covers COM stack jitter in the input event models as well as interference from small non-synchronous control data. In this context, without a direct sample on resource overlapping, M should assure that all frames of an ADAS sample are transmitted before the start of the next one.

Comments on routing: Besides the offset configuration, the second configuration option of a stream is its route. In automotive systems, Ethernet network topologies are very simple, hence, routing strategies are not of relevance. A ring topology, as used in this work, is even one of the more complex setups, however, there are at most two routes for any connection between two NICs. In this work, we take always the shortest route. If 3 switches have to be passed (e.g., NIC 7 to NIC2 in Figure 3), we choose randomly one of the two routes with equal size.

5 SYNCHRONOUS RESPONSE TIME ANALYSIS

In this section, based on the system model of Section 3, we present our straightforward response time analysis. On the one hand, it enables us to prove the expectation of very low synchronous sample data latencies that follow the resolved interference between ADAS streams. On the other hand, it let us determine the maximum impact of small but high-priority control data. While sporadic analysis for high-priority control data is well known, we focus on the analysis of WCRTs of synchronous ADAS data transmissions that share Ethernet output port resources with this non-synchronous control data.

As we have specified input event models and event model propagation, we follow the global convergence considerations of CPA [10]. In specific, after performing the (synchronous) response time analysis of all *local* output port resources, the derived output event models are propagated along the Ethernet stream task chains. This is called the *global* analysis step, whereby convergence of the analysis is reached in case the input event models of tasks do not change anymore after the event model propagation step. Otherwise, the task analysis on local output port resources that have changed input models is repeated.

5.1 The Maximum Synchronous Workload Schedule

To determine the WCRTs of synchronous tasks in local resource analysis, a corresponding analysis has to start with a worst-case consideration. In a sporadic design, this is the critical instant. For such, based on an idle resource state, the most intense workload arrival follows the minimum distance functions of tasks on that resource. As depicted in Figure 4 it defines the minimum time distance between a number of n consecutive events. In contrast, in the synchronous analysis, a comparable critical instant would rather be a badly configured system, where all ADAS streams transmit their data at the same time in the hyperperiod. To avoid or at least strongly reduce such

Table 2. A Synchronous Example System of Three ADAS Tasks Scheduled on the Same Output Port Resource

task	d_c	J_c	φ_c	C_i^+	N_c
τ_1	100 µs	200 µs	0 µs	100 µs	4
τ_2	100 µs	200 µs	300 µs	100 µs	4
τ_3	200 µs	300 µs	900 µs	100 µs	2

We assume that they share the same sample period, which is larger than the processing time of a burst.

analysis pessimism between synchronous sample transmissions, in order to benefit from the non-overlapping offset configuration from Section 4, we have to consider only the maximum load that can be persistent on an output port resource at a *certain time interval in the hyperperiod*. As a synchronous worst-case assumption for the synchronous ADAS load, we specify the *maximum synchronous workload schedule* $L^+_{\mathrm{sy},r}(t)$:

Definition 5.1. The **maximum synchronous workload schedule** $L^+_{\mathrm{sy},r}(t)$ of a resource r describes the maximum persistent workload at a certain time $0 \leq t < T_H$ in the hyperperiod, that origins from any synchronously scheduled event source.

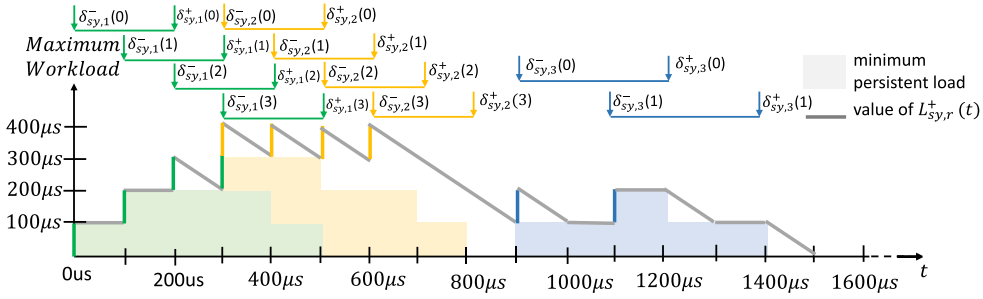
The idea is, that based from a known maximum state of unprocessed frame load at any t in T_H, which is represented by $L^+_{\mathrm{sy},r}(t)$, we can open a busy window to analyze the WCRT of a frame that is pending for transmission at that time. In specific, as depicted in Figure 5, the WCRT of the n-th frame of a synchronous ADAS stream starts from its latest possible arrival time $\delta^+_{\mathrm{sy},i}(n)$ on an output port. The maximum interfering frame workload to be processed at that time is then $L^+_{\mathrm{sy},r}(\delta^+_{\mathrm{sy},i}(n))$.

The construction of $L^+_{\mathrm{sy},r}(t)$ is described in the following and visualized based on the example setup of three ADAS tasks summarized in Table 2. The worst-case assumption is that synchronous load arrives *as early as possible* corresponding to $\delta^-_{\mathrm{sy},i}(n)$ but is allowed to be processed *as late as possible* corresponding to $\delta^+_{\mathrm{sy},i}(n)$. This maximizes the time that workload persists in an output port resource and, hence, the maximum workload that persists at a certain time in the hyperperiod. Moreover, as all sample periods T_c are homogeneous within the hyperperiod, we can expect a repeating behavior of $L^+_{\mathrm{sy},r}(t)$ each T_H. Following this consideration, the load schedule for the example setup of Table 2, which is depicted for an exemplary part of the hyperperiod in Figure 8(a), can be constructed as follows:

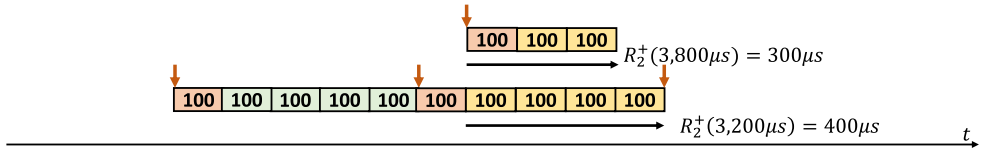
(1) **Persistent workload:** The n-th activation of a task $\tau_{i,c}$ can occur at any time in its activation interval, which is between $\delta^-_{\mathrm{sy},i}(n)$ and $\delta^+_{\mathrm{sy},i}(n)$. Hence, the corresponding task's WCCT C_i^+ has to be considered as a persistent workload during that interval. Consequently, the lower bound of $L^+_{\mathrm{sy},r}$ at time t is the sum of the workload of all n-th frame transmissions of all tasks on that resource whose activation interval includes t. This persistent workload, which has to be considered as a lower bound of $L^+_{\mathrm{sy},r}(t)$, is depicted as the shaded area in Figure 8(a).

(2) **Load arrival**: The construction starts at $t = 0$ in the hyperperiod. Each $\delta^-_{\mathrm{sy},i}(n)$ or $\delta^+_{\mathrm{sy},i}(n)$, either load arrives or is allowed to be processed. Hence, we can iterate along all $\delta^{-/+}_{\mathrm{sy},i}(n)$ mod T_H values in temporal order. Considering the current maximum workload value we add C_i^+ to it at any $\delta^-_{\mathrm{sy},i}(n)$.

(3) **Load processing:** We process load between two consecutive $\delta^{-/+}_{\mathrm{sy},i}(n)$ values of any tasks. Following the above consideration of persistent workload, $L^+_{\mathrm{sy},r}(t)$ can not drop below the maximum persistent workload value (shaded area).



(a) The maximum synchronous workload schedule for one ADAS sensor data sample transmission of each task from Table 2 in the hyperperiod.



(b) Starting the critical instant of a sporadic control task τ_4 at $t = 200\,\mu s$ leads to a larger response time value of $R^+_2(3, 200\,\mu s)$ than starting directly at $t = \delta^+_{sy,2}(3) = 800\,\mu s$. Here, the task τ_4 has a minimum distance between the first two activations of 500 µs.

Fig. 8. The maximum synchronous workload schedule and the effect of different evaluation times t in the WCRT analysis of a synchronous task activation under sporadic load interference.

(4) **Stopping condition:** The calculation stops at the first time $t \mod T_H$ reaches an already passed time value and for which $L^+_{sy,r}(t)$ results in the same workload as in the previous hyperperiod. In that case, based on the repeating behavior of each hyperperiod, the above steps will lead to the same $L^+_{sy,r}(t)$ values. Note that if the resource is not overloaded solely by synchronous load, the construction always terminates at the latest after the second hyperperiod. Otherwise, the load accumulates rather than being processed through T_H.

5.2 Synchronous Worst-case Response Time Analysis

Based on the sporadic and synchronous task event models (cf. Sections 3.2 and 3.3), the maximum synchronous workload schedule (cf. Section 5.1), as well as the scheduling policy (cf. Section 3.4), we now formulate the local response time analysis for synchronous tasks. As a result, we obtain the WCRT $R^+_i(n)$ of the n-th task activation representing the n-th frame communication of a task on a local output port resource. This WCRT value is then used for the construction of the output event models (cf. Equation (6)), which are propagated along the task chains in the global analysis step of CPA. Moreover, after event model convergence, WCRT values are used to calculate the WCRTs of whole synchronous sample communications.

As depicted in Figure 5, we have to start $R^+_i(n)$ analysis from $\delta^+_{sy,i}(n)$, as it represents the latest possible activation time in the hyperperiod T_H. This way, the propagated arrival time interval in the hyperperiod, in which the frame arrives at the next switch output port along its task chain, is widened as much as possible, representing the worst-case. For same-priority synchronous load, the FIFO scheduling of IEEE 802.1Q precludes any blocking of frames that arrive later than $\delta^+_{sy,i}(n)$. Hence, $L^+_{sy,r}(\delta^+_{sy,i}(n))$ gives us the maximum load value for synchronous ADAS-related load to consider in $R^+_i(n)$. However, this is not the case for sporadic load. First, higher-priority load, that arrives after $\delta^+_{sy,i}(n)$ interferes with a task's n-th activation and, hence, opens a *maximum busy*

window that ends until all interfering higher and same priority load has been processed. Second, worst-case response time results $R_i^+(n,t)$ of the n-th frame starting from $\delta^+_{\text{sy},i}(n)$ change with the placement time t of the sporadic critical instant in the hyperperiod. This is shown in the example depicted in Figure 8(b). The sporadic critical instant for the task τ_4 leads to a higher WCRT for $R_2^+(3, 200\,\mu\text{s})$ if placed at $t = 200\,\mu\text{s}$ than directly at $t = \delta^+_{\text{sy},2}(3) = 800\,\mu\text{s}$. In other words, the overshoot w.r.t. the combined sporadic/synchronous interference of the maximum busy window $B_i^+(n,t)$ towards the reference point $\delta^+_{\text{sy},2}(3)$ is larger when starting at $t = 200\,\mu\text{s}$. Evaluating the WCRT $R_i^+(n,t)$ for a specific $t \in T_H$ is equals to that overshoot of the maximum busy window $B_i^+(n,t)$ from t to $\delta^+_{\text{sy},i}(n)$:

$$R_i^+(n,t) = t + B_i^+(n,t) - \delta^+_{\text{sy},i}(n) + C_i^+ \tag{7}$$

Due to SPNP scheduling, we add C_i^+, representing the n-th frame, outside the maximum busy window, because it cannot be disturbed after it starts processing. For determining the overall task's n-th WCRT $R_i^+(n)$, we have to determine the $t \in T_H$ that maximizes Equation (7):

$$R_i^+(n) = \max_{0 \le t < T_H} \{R_i^+(n,t)\} \tag{8}$$

To find the $t \in T_H$ that maximizes $R_i^+(n)$ it is sufficient to search at points in time where the most interference queues up instead of being processed, because this also leads to the maximum queuing of sporadic load. In $L^+_{\text{sy},r}(t)$, this is given at $t = \delta^+_{\text{sy},i}(n)$ values as the load is given free for processing thereafter. A more detailed discussion on the complexity of that search is provided in Section 5.4.

The next step is to calculate the maximum busy window which includes lower, same as well as higher priority load:

$$B_i^+(n,t) = \underbrace{I_i^{\text{LP}}}_{\substack{\textbf{I.}\ \text{Lower}\\ \text{priority blocking}}} + \underbrace{I_i^{\text{SP}}(n,t)}_{\substack{\textbf{II.}\ \text{Same priority}\\ \text{synchronous blocking}}} + \underbrace{I_i^{\text{HP}}(n,t)}_{\substack{\textbf{III.}\ \text{Higher priority}\\ \text{sporadic blocking}}} \tag{9}$$

First, under non-preemptive SPNP scheduling, we have to account for lower priority *lp* interference I_i^{LP} to a task τ_i, represented by the maximum WCCT of any lower priority task on the same resource:

$$I_i^{\text{LP}} = \max_{j \in lp(i)} \{C_j^+\} \tag{10}$$

Within Equation (9) the maximum interfering load from same-priority *sp* communication tasks $I_i^{\text{SP}}(n,t)$, starting from t, is as follows:

$$\begin{aligned} I_i^{\text{SP}}(n,t) = L^+_{\text{sy},r}(t) - C_i^+ &+ \underbrace{\left(\sum_{j \in sp(i)} (\eta^-_{\text{sy},j}(\delta^+_{\text{sy},i}(n)) - \eta^-_{\text{sy},j}(t)) \cdot C_j^+ \right)}_{\text{Synchronous load arrival until } \delta^+_{\text{sy},i}(n)} \\ &- \underbrace{(\eta^-_{\text{sy},i}(\delta^+_{\text{sy},i}(n)) - \eta^-_{\text{sy},i}(\delta^-_{\text{sy},i}(n))) \cdot C_i^+}_{\text{Remove overpassing of same task}} \end{aligned} \tag{11}$$

The equation starts with the maximum persistent synchronous workload at $L^+_{\text{sy},r}(t)$. From that, we have to subtract once the communication workload of τ_i, as it has already been added outside the busy window in Equation (7). Next, all additional synchronous load that can arrive from t until $\delta^+_{\text{sy},i}(n)$ is added for all same priority tasks j. Therefore, the *lower synchronous event arrival time function* $\eta^-_{\text{sy},i}(t)$ is evaluated at both times. The difference is the maximum number of task activation in that time interval and is multiplied by the j task's WCCT. Last, as overtaking frames

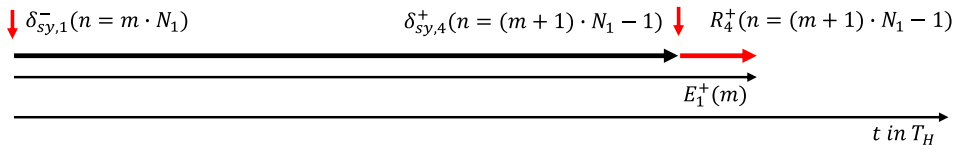


Fig. 9. Exemplary worst-case response time analysis of the m-th synchronous sample transmission of task chain $c = 1$ corresponding to Figure 3.

within a task chain is not possible, we subtract activation of the i-th task itself, which arrive after $\delta^-_{\text{sy},i}(n)$ until $\delta^+_{\text{sy},i}(n)$.

Last, the maximum higher-priority *hp* sporadic load can delay the transmission of synchronous frames at any time. This spans a maximum busy window $B^+_i(n,t)$ for n at t:

$$I^{\text{HP}}_i(n,t) = \sum_{j \in hp(i)} \eta^+_{\text{sp},j}(B^+_i(n,t) + \epsilon) \cdot C^+_j \tag{12}$$

By including the busy window into $I^{\text{HP}}_i(n,t)$ we get a fix-point problem. It can be solved by iteration, starting from $B^+_i(n,t) = 0$, as the load over time is a monotonously increasing function. In SPNP we have to ensure that the non-preemptable frame of task i, which is under investigation and added outsite the busy window (cf. Equation (7)), is only started in case the busy window is empty *and* there is no newly arriving load. This is achieved by adding the minimum scheduling granularity ϵ to the busy window.

5.3 Worst-case Response Time Analysis of Sample Transmissions

Based on the calculation of the worst-case response times corresponding to the individual n-th frame transmissions of tasks, we can now analyze the worst-case response time E^+_c of an end-to-end sample level transmission that corresponds to a certain *task chain* c. For the m-th sample transmission in the hyperperiod it can be calculated as follows:

$$E^+_c(m) = \underbrace{\delta^+_{\text{sy,last}(c)}((m+1) \cdot N_c - 1) + R^+_{\text{last}(c)}((m+1) \cdot N_c - 1)}_{\text{Latest time the sample transmission ends}} - \underbrace{\delta^-_{\text{sy, first}(c)}(m \cdot N_c)}_{\substack{\text{Synchrous start} \\ \text{of sample transmission}}} \tag{13}$$

Exemplary, for chain 1 of Figure 3 the principle is depicted in Figure 9. The m-th sample transmission starts with the transmission of the first task $\delta^-_{\text{sy},i=\text{first}(c)}(n)$ with the first frame $n = m \cdot N_c$. It ends after the worst-case response time of the last frame of that m-th sample transmission, which is at $n = (m+1) \cdot N_c - 1$ starting from $\delta^+_{\text{sy},i=\text{last}(c)}(n)$. The end-to-end worst-case response time of a task chain-related sample transmission E^+_c is the maximum $E^+_c(m)$ value over all $m \in \{0, 1, \ldots, \frac{T_H}{T_c} - 1\}$ sample transmissions in the hyperperiod.

For later comparison with the state-of-the-art, the worst-case response time of a task chain for the sporadic timing analysis $E^+_{\text{sp},c}$, which is based on *sporadic arrival curves*, can be determined for a sample transmission as follows:

$$E^+_{\text{sp},c} = \delta^+_{\text{sp,first}(c)}(N_c) + \sum_{i \in S_c} R^+_{\text{sp},i} \tag{14}$$

Starting from the first task in the chain c, we take the maximum distance $\delta^+_{\text{sp,first}(c)}$ between the injection of all N_c sample frames and add the sum of the worst-case response time $R^+_{\text{sp},i}$ of tasks along the chain. For further details on non-synchronous task response time analysis for sporadic Ethernet communication please refer to the works of [10] and [31].

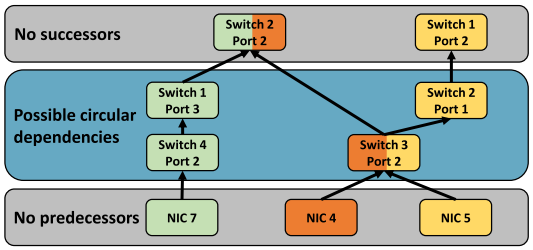


Fig. 10. Event propagation along synchronous task chains from Figure 3 abstracted on output port level. As there are no circular dependencies, each synchronous task has only to be analyzed once.

5.4 Synchronous Analysis Complexity

The complexity of the analysis depends on two dimensions. The first is the number of tasks to be (re)evaluated in case their input model change and the second is the number of frames to be analyzed on one resource.

Number of task evaluations: Our analysis first evaluates the highest priority tasks, which are representing the small control data. As it is not interfered by (does not depend on) the larger ADAS data, it remains stable after convergence. If compared to synchronous ADAS analysis, sporadic Ethernet analysis runtime is negligible.

In the synchronous analysis step, the converged control data event models then only serve as a stable source of interference on the output ports. In an Ethernet ring topology, circular dependencies can arise. This might lead to very pessimistic results, or even non-convergence of the analysis, especially in cases where the workload is very high, as it is with our ADAS streams. However, as we will show, such circular ADAS dependencies are very unlikely and are rather a result of bad design. Figure 10 depicts the dependencies of the example setup shown in Figure 3. As ADAS streams share the same priority, we can abstract dependency relations on resource level. In detail, if an ADAS task chain propagates its output model to another output port following the chain's path, all ADAS tasks on that resource have to be reanalyzed, as the input model of an interfering stream has changed. A circular dependency is represented by circles along that dependency relation and might result in non-convergence. However, in the shown example, there is no such dependency. Thus, if we analyze the output ports with their corresponding allocated tasks along their dependencies, **each task has to be analyzed only once**. A more general perspective on the avoidance of circular dependencies is shown in Figure 11. It shows the full dependency graph under the assumption that only the NICs connected to switch 4 are not a target for ADAS data reception. This reflects common design approaches, as sensor data processing is usually performed on a few instead of all network nodes. As one can see, there are no circular dependencies under that condition. Hence, independent from our concrete analysis, we can show this way that the synchronous design approach does not have a problem with circular dependencies in general unless there is a very unfavorable design.

Local analysis complexity: The synchronous design approach comes at the cost to analyze each frame individually in the hyperperiod. Therefore, WCRT analysis demands checking the critical instant referring to higher priority sporadic load at any $\delta^+_{\mathrm{sy},i}(n)$ corresponding to all tasks on the same resource. This leads to the complexity class of $O(n^2)$ with the number of frames per resource. While the local analysis of tasks should demonstrate the benefits of the synchronous design approach, the analysis performance is not key to this work. However, future work can further exploit concepts that analyze WCRTs of multi-frame sample transmission as a hole. This could significantly reduce complexity to the number of samples instead of frames, making complexity

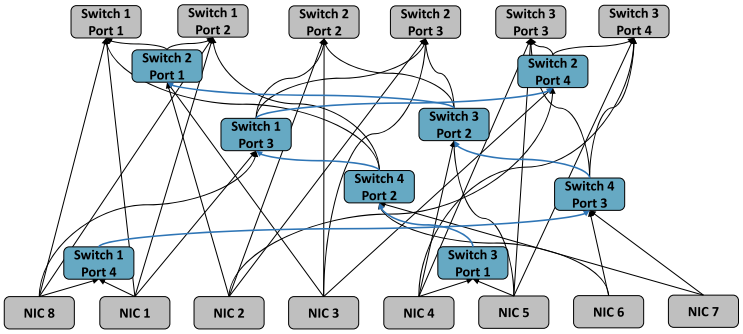


Fig. 11. Full dependency relations between resources of Figure 3. Only NIC 6 and 7 do not serve as end nodes for ADAS streams. Hence, ports 1 and 4 of Switch 4 are not used. Blue lines represent dependencies between switches that are relevant for circular dependencies.

independent of sample size and enabling efficient analysis of higher Ethernet data rates. Here, a combination of the *multiple event busy time* analysis approach from [30] together with offsets may be a sophisticating further extension. A straightforward improvement to the presented approach can be achieved by analyzing independent tasks along their dependencies in parallel. Moreover, as deployed in our work, we only have to consider reference times t to the evaluation of $R_i^+(n)$, for which the maximum busy window $B_i^+(n,t)$ at least reaches $\delta_{\text{sy},i}^+(n)$: $t + B_i^+(n,t) \geq \delta_{\text{sy},i}^+(n)$. Otherwise, $R_i^+(n,t)$ would become less than the tasks WCCT (cf. Equation (7)). This bounds the candidate search to the burst that contains $\delta_{\text{sy},i}^+(n)$.

6 EVALUATION

We start this section by describing our experimental setup as well as the corresponding parameterization. Therefore, we base our offset configuration on the considerations from Section 4, which should be sufficient to demonstrate the overall benefits of our synchronous approach. It follows an evaluation of the sample worst-case response times based on both analysis and simulation. To show the benefits of our synchronous design approach, we compare it to the sporadic design in both cases. We conclude the evaluation with an additional comparative discussion on TAS.

6.1 Experimental Setup

The Ethernet topology used in the evaluation is the ring topology from the system model depicted in Figure 3. All control and ADAS stream parameters that are described in the following are summarized in Table 3. The network data rate is given with 100 Mbps. We consider that all ADAS frames are transmitted with a total size of 1500 B, whereby – following current automotive use cases – control data is of size 512 B [26]. For simplicity, we exclude considerations of protocol overhead to payload ratio in the frames and rather focus on the total number that is necessary to transmit a sample. As the Ethernet network does not care about the content of the data, this simplification does not impact our analytical and simulative results.

Control stream configuration: For each setup, we deploy *20 control streams* with random routes between any two different NICs. Such streams send relatively large control data frames of 512 B, with a period of 50 ms.

Synchronous ADAS stream configuration: While in typical automotive setups ADAS streams only target a limited number of end nodes, we let ADAS streams start at 6 NICs and end

Table 3. Overview of Basic Parameter Configurations for Control and ADAS Streams

	Parameter	Value
Overall configuration	Hyperperiod T_H	1 s
	Data rate R_r	100 Mbps
	Jitter J_c	500 µs
Control stream configuration	Number of control streams	20
	Sample period	50 ms
	Sample size = Frame size	512 B
	Frames per sample N_c	1
ADAS stream configuration	Number of ADAS streams	{4,8,12,16,20,24}
	Sample period	100 ms
	Sample size	90 kB
	Frame size S_i^+	1500 B
	Frames per sample N_c	60
	Frame distance d_c	120 µs
	Sample robustness margin M	1.5 ms

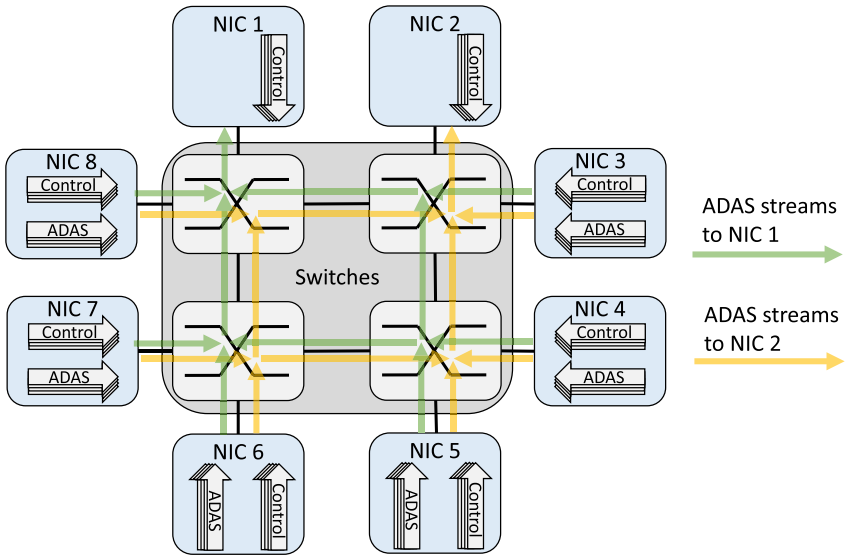


Fig. 12. Different colors represent the taken ADAS stream routes towards either NIC 1 or NIC 2. The underlying ring topology is the one depicted in Figure 3.

at one of the two remaining NICs, which are NIC 1 and NIC 2. Following the consideration from Section 5.4, there cannot be circular dependencies on ADAS stream level. All possible static routes are depicted in Figure 12, whereby each color corresponds to one end NIC. For different experimental setups, we increase the number of ADAS streams by 4 each round, leading to a range from 4 up to 24 streams. Each round 2 streams are routed to NIC 1 and 2 streams are routed to NIC 2. ADAS streams have a sample size of 90 kB that is divided to 60 frames and sent with a sample period of 100 ms. We configure the jitter to 500 µs [16]. Based on random routes, the offset is configured following Section 4. Here, to avoid ADAS stream overlapping due to control data interference as well as COM stack jitter, a robustness margin M of 1.5 ms is deployed.

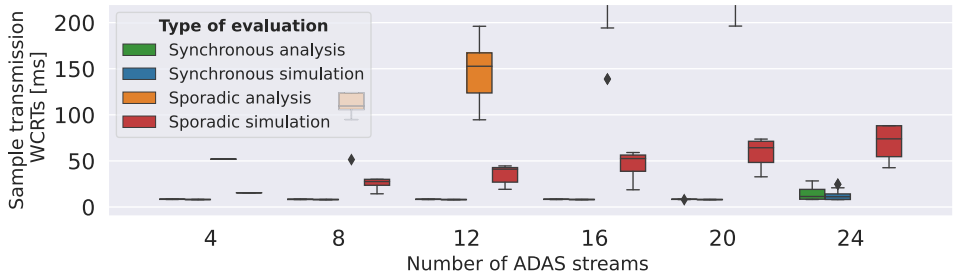


Fig. 13. Worst-case response times of ADAS streams for analysis and simulation both synchronous as well as sporadic. For 24 ADAS streams, the lowest value of the sporadic analysis exceeds the depicted range of sample transmission WCRTs.

Sporadic ADAS stream configuration: For comparison, sporadic ADAS streams take the same routes and the same parameterization. The only difference is the use of non-synchronized sporadic event models as described in Section 3.2.

Simulative setup: For simulation, we deployed the openly available TSN network simulator [22], which is based on the popular Objective Modular Network Testbed in C++ (OMNeT++) network event simulator [5]. It directly represents our presented task chain to resource mapping from Section 3. Moreover, the simulation processes incoming frames that represent task activations following IEEE 802.1Q -based SPNP and FIFO scheduling from Section 3.4. The initial task chain activation of synchronous streams follows Equations (3) and (4). Non-synchronous sporadic activation is initially started simultaneously while varying towards the next activation with the assigned jitter. This reflects current CEC design approaches and is close to the critical instant. We let the simulation run for 600 s, which provides sufficiently expressive results.

Analytical setup: While the synchronous local resource analysis is based on Section 5, the sporadic analysis is represented by the state-of-the-art IEEE 802.1Q CPA-based analysis from [31]. Both are integrated into the global event model propagation step of CPA. Therefore, we used the openly available framework pycpa [10].

6.1.1 Overall Latency Evaluation. To get a general overview of the results from the different configurations, Figure 13 shows the absolute WCRTs of all individual streams. First, we see that the sporadic analysis results significantly degrade with a higher number of deployed ADAS streams. Therefore, it becomes apparent that the lack of synchronization of large data sample transmissions leads to unacceptably high worst-case latencies. Hence, the analysis seems to deliver completely unusable results, in which even the FIFO exploitation from [31] is not relevant in contrast to the load being too high. Note that for 24 ADAS streams all worst-case response times are clearly above 200 ms, so we have not shown them in the diagram for a better overview. Next, the sporadic simulation generates the second worst results. For that, the degradation effects along higher stream numbers seem to increase linearly to up to 88.2 ms for 24 ADAS streams, which is intuitive as a sample can wait to up to that amount of other interfering samples once. Last, in contrast, the synchronous simulation and analysis deliver constant WCRTs at low levels even to up to 20 ADAS streams. This states the goals of our synchronous transmission approach: (1) to improve the latency by removing interference between ADAS streams. (2) to tolerate small control data at the highest-priorty by also introducing robustness margins between the ADAS stream transmissions that are *placed* by an offset into the hyperperiod.

Considering the synchronous analysis complexity, the computation of 24 streams took $\approx$ 2.7 hours. As discussed in Section 5.4, this value can be significantly decreased by e.g., parallel

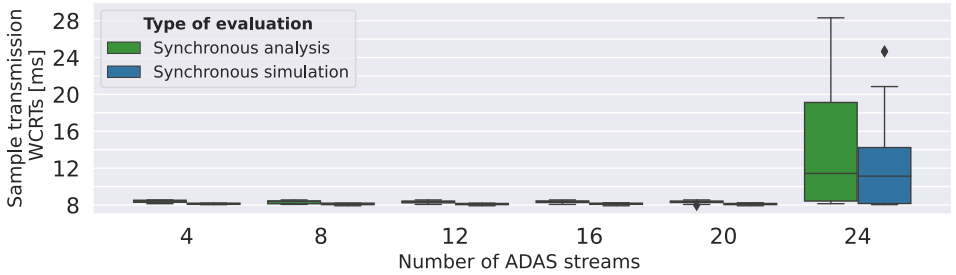


Fig. 14. Analytical and simulative worst-case response times of synchronous ADAS streams.

task evaluation with multi-threading or advanced analytical procedures. However, the focus of this work is to show the advantages of the synchronous design over non-synchronous state-of-the-art approaches. In this context, the analysis – together with the simulation – is sufficient to confirm that claim.

6.2 Synchronous Latency Evaluation and Utilization

Figure 14 provides a more detailed view on the synchronous results. We see that the WCRTs are stable until an ADAS stream number of 20 and degrade for 24 ADAS streams as there is *no free slot* in T_H available any more on all passed NIC and switch output ports (cf. Section 4). The results up to 20 ADAS streams support our assumptions that independence between large ADAS streams can be achieved based on a synchronized network transmission of data samples that reduces the jitter to the one of the COM stack and avoids the critical instant. Thereby, for up to 20 ADAS streams, the analytical as well as the simulative upper WCRTs of either 8.55 ms or 8.25 ms are close to their theoretical non-interfered WCRTs of 8.18 ms when considering at most 4 hops and full jitter. Interestingly, the highest WCRT of any ADAS stream resulting from the synchronous analysis (even for 24 ADAS streams) is still lower than the lowest WCRT from any ADAS stream following the sporadic simulation. Hence, even a limited overlap of synchronous streams, that can originate from very high utilization, still creates significantly less intense frame burst interference than the sporadic design. In correspondence, as depicted in Figure 15, the utilization that can be achieved without any interference degradation between 20 ADAS streams is 72%. With limited degradation at 24 streams a utilization of up to 88% could be reached. For 20 ADAS streams, the remaining 28% up to 100% can be used without noteworthy interference to lower priority traffic. For the given synchronized sample periodicity of 100 ms, a lower priority task has at maximum to wait that 100 ms to get the remaining data rate. While this should be sufficient even for entertainment applications, more realistically, the remaining data rate is much more evenly divided over T_c considering the robustness margin M between sample transmissions. In both cases, lower priority applications can easily use the total remaining data rate. Therefore, in terms of non-safety-critical soft-deadlines, it is sufficient to rely on simulation, as done by the industry.

In summary, our synchronous event model and analysis approach achieves results that are close to the synchronous simulative results as well as their theoretical optimums. Therefore, the synchronous approach is a significant improvement over sporadic system design. Moreover, the stability of WCRTs proves the independence between ADAS streams even under the presence of small interfering control messages. Since [6] could show that automotive software, in particular, has strongly varying execution times, resynchronization at the network boundaries is of great advantage. Otherwise, without active resynchronization, data communication can not be expected to have predictable communication times following the synchronous start of

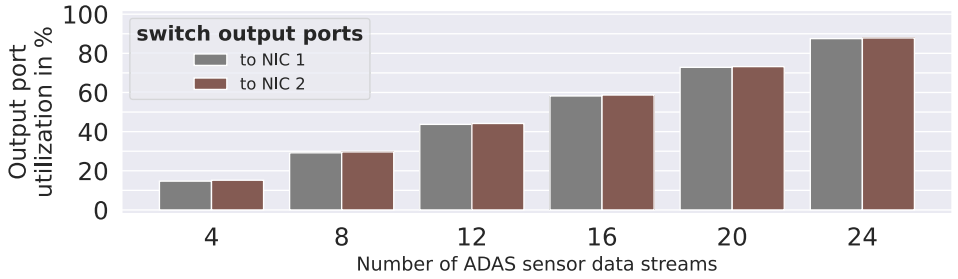


Fig. 15. The link utilization on the switch output ports leading from Switch 1 *to NIC 1* and from Switch 2 *to NIC 2*. Utilization is independent of sporadic or synchronous design.

CECs. This, therefore, goes beyond the initial transmission of raw sensor data and also includes non-negligible communication loads along the autonomous driving CEC. As our work has considered a specific configuration, the principles apply generally to any system setup. This makes our synchronous system design and analysis approach applicable to the increasing sensor data sizes to be transmitted that follow higher available Ethernet data rates. The only prerequisite is to guarantee an upper jitter, which can be achieved by timed programming and can be further supported by recent developments in real-time capable COM stack design itself [16].

6.3 TAS Comparison

In the light of the evaluation results, we can now compare the potential benefits when using TAS instead of the proposed solution. Since TAS complicates the protocol and is not workload preserving due to the necessary guard bands and the non-preemptable frame overhead [2], there should be substantial benefits in required throughput, latency or stream isolation in at least one of the communication classes. Originating in the need for small control packet communication in industrial setups, TAS deploys a TDMA cycle of the periodicity T_C. In each cycle, slots can be assigned to different Ethernet priorities. Synchronizing such cycles over all hops on a given route should minimize the upper packet latencies. In the following we discuss the configuration of the 3 main priority classes in the context of safety-critical latency requirements in reference to [26].

(1) Priority 1 (AVB class A): Safety-relevant control and network control data.
(2) Priority 2 (AVB class B): Safety-relevant media data.
(3) Priority 3 and lower (AVB best effort): Other non-safety critical data.

Configuration of high-priority class 1: Following the main TAS motivation, a control frame should be transmitted with a latency of about one millisecond. However, if a non-synchronized frame transmission misses the TAS slot, it has to wait until the next one in the next TAS cycle. As a consequence, the TAS cycle period T_C should not be larger than that 1 ms. However, in terms of overall load considerations, the subordinate event period of the arrival curve that represents the control data transmission is much larger than 1 ms. In this case, most of the pre-allocated control data slots in each TAS cycle are left idle and cannot be used by lower priority frames, such as safety-relevant ADAS media data. As a consequence, *TAS is not workload-preserving*. For 512 B frames [26] that slot time has to be at least $\approx$ 40 µs per cycle. The guard band to protect the slot adds up the duration of a preemptable Ethernet frame of 143 bytes, corresponding to ca. 12 µs [32]. Together this blocks 5.2% of overall transmission time.

In our example and evaluation without TAS, the worst-case response time for high-priority control data can also be bounded below 1 ms as requested in the Automotive TSN profile [25],

even without frame preemption. Given a number of hops N_{hop}, we have to consider 120 µs of low-priority blocking interference I^{LP} per hop. Thereupon, a maximum number of other crossing control streams N_{cs}^{+} add up an additional blocking of $C_{\text{cs}}^{+} = 40\,\mu\text{s}$ each. If the control stream under investigation crosses a high number of other streams $N_{\text{cs}} = 10$, the worst-case response time E_{cs}^{+} of that control data stream can be bounded as follows:

$$E_{\text{cs}}^{+} = N_{\text{hop}} \cdot I^{\text{LP}} + N_{\text{cs}}^{+} \cdot C_{\text{cs}}^{+} = 4 \cdot 120\,\mu\text{s} + 10 \cdot 40\,\mu\text{s} = 880\,\mu\text{s} \tag{15}$$

With these data, unless the latency requirements are substantially shorter or the network is much larger, there is no obvious need to resort to TAS for class 1 traffic, as is described in the automotive use case for zonal architectures [26]. If needed, the lower priority classes could use preemptable frames, such that the blocking time in Equation (15) could be reduced from 120 µs to 12 µs, as in the case of TAS. In the following, we show that, as a consequence, the safety-relevant ADAS media data streams would be seriously degraded if being forced into the TAS cycle.

Implications on priority class 2: On the lower priority 2, ADAS sample data transmissions, the main interest of the paper, have to be managed besides the already allocated priority 1 control data slots. Case 1: Let us first assume that class 1 is not using TAS, as in our paper. Then, in our approach, the safety-relevant media data packets of a class 2 sample are transmitted back-to-back, only interfered by class 1 control data frames. This is the minimum achievable latency per link, and it is workload-preserving as all unused link capacity is available for lower priorities. Hence, there is no advantage in using TAS for link latency reduction. What could be reduced is the blocking time per link. For a fair comparison, we may assume the use of preemptable lower-priority frames. Then, the maximum blocking time per hop is 12 µs, corresponding to less than 50 µs for 4 hops, and less than 1% increase in total sample transfer time in our evaluations. That small latency overhead is outweighed by the benefits of workload preservation. Even if we avoided the overhead of lower-priority preemptable frames, the total latency increase would be less than 7%. Case 2: Let us, secondly, assume that class 1 is using TAS, but class 2 is not. Then, the interference by class 1 on class 2 is growing by empty slots and guard bands, beyond the interference in case 1, leading to higher sample latency and link load than in case 1. Case 3: Let us, finally, assume that both class 1 and class 2 are using TAS, possibly sharing slots. Then, the maximum potential benefit is as summarized in case 1, i.e., less than 1% lower latency, provided the far more complex scheduling problem could be solved with perfect adjustment of link slots and the resulting hierarchical scheduling problem. That is quite unlikely in general.

Implications on lower priorities: All lower-priority traffic can only win in available bandwidth if TAS is avoided.

To summarize, in the discussed zonal architecture, TAS can only be beneficial for class 1 traffic, but is only needed if the latency requirements are <1 ms, which is, however, not required in the IEEE 802.1 automotive profile. We may therefore conclude that the extra complexity of TAS will not be justified in the investigated use case of a zonal architecture. The conclusion might be different in a significantly different network architecture with many more hops and a different load profile. This is beyond this paper.

7 CONCLUSION

In this work, we presented a design approach for the communication of large sensor data samples that avoids inter-sample interference by synchronizing their transmission times in a global hyperperiod. This principle is based on upcoming synchronization techniques such as SL-LET as well as real-time COM stack developments and demonstrates the benefits of targeting specific transmission times of large sample data to the network. The exclusive focus on synchronous data injection in terms of network control allows for a significant reduction in design complexity compared to

the much more challenging synchronization and transmission procedures in communication technologies such as TSN. Without delaying the small safety-relevant and latency-critical control data messages, we could show that the transmissions of large ADAS sensor data samples can be decoupled from each other in order to bind their transmission latencies at a low level. Thereby, the latency results of our supportive analysis are very close to the simulated results as well as to the theoretical minimum. In contrast, without synchronization, collisions on sensor data sample level lead to unacceptable high communication latencies of that time-critical larger media data, which is also safety-relevant. Thus, this work clearly shows that it is reasonable to remove the complexity from the network hardware and focus on the synchronization of sample input times instead. The resulting reductions in worst-case communication latencies, even at limited sample transmission overlaps, make it reasonable to adjust time markers in the software control of CECs in order to improve on their end-to-end latency guarantees.

REFERENCES

[1] 2015. IEEE approved draft standard for local and metropolitan area networks - media access control (MAC) bridges and virtual bridged local area networks amendment: Enhancements for scheduled traffic. *IEEE P802.1Qbv/D3.1, September 2015* (2015), 1–52.

[2] 2016. IEEE standard for ethernet amendment 5: Specification and management parameters for interspersing express traffic. *IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as Amended by IEEE Std 802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016)* (2016), 1–58. https://doi.org/10.1109/IEEESTD.2016.7592835

[3] 2018. IEEE standard for local and metropolitan area network–bridges and bridged networks. *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)* (2018), 1–1993. https://doi.org/10.1109/IEEESTD.2018.8403927

[4] 2021. IEC/IEEE international standard - precision clock synchronization protocol for networked measurement and control systems. *IEC/IEEE 61588-2021* (2021), 1–504. https://doi.org/10.1109/IEEESTD.2021.9456762

[5] 2022. OMNet++ Discrete Event Simulator. https://omnetpp.org/

[6] Miguel Alcon, Hamid Tabani, Leonidas Kosmidis, Enrico Mezzetti, Jaume Abella, and Francisco J. Cazorla. 2020. Timing of autonomous driving software: Problem analysis and prospects for future solutions. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'20)*. IEEE, 267–280.

[7] Anna Arestova, Kai-Steffen Jens Hielscher, and Reinhard German. 2021. Simulative evaluation of the TSN mechanisms time-aware shaper and frame preemption and their suitability for industrial use cases. In *2021 IFIP Networking Conference (IFIP Networking'21)*. IEEE, 1–6.

[8] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. 2021. Time-sensitive networking in automotive embedded systems: State of the art and research opportunities. *Journal of Systems Architecture* 117 (2021), 102137. https://doi.org/10.1016/j.sysarc.2021.102137

[9] Hugo Daigmorte and Marc Boyer. 2016. Traversal time for weakly synchronized CAN bus. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. 35–44.

[10] Jonas Diemer, Philip Axer, and Rolf Ernst. 2012. Compositional performance analysis in python with pycpa. *Proc. of WATERS* (2012), 46.

[11] Jonas Diemer, Jonas Rox, and Rolf Ernst. 2012. Modeling of Ethernet AVB networks for worst-case timing analysis. *IFAC Proceedings Volumes* 45, 2 (2012), 848–853.

[12] Jonas Diemer, Daniel Thiele, and Rolf Ernst. 2012. Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching. In *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*. 1–10. https://doi.org/10.1109/SIES.2012.6356564

[13] Rolf Ernst, Leonie Ahrendts, and Kai-Björn Gemlau. 2018. System level LET: Mastering cause-effect chains in distributed systems. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 4084–4089.

[14] Emilia Farcas, Claudiu Farcas, Wolfgang Pree, and Josef Templ. 2005. Transparent distribution of real-time components based on logical execution time. In *Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*. 31–39.

[15] Kai-Björn Gemlau, Hermann v Hasseln, and Rolf Ernst. 2022. Industry-track: System-level logical execution time for automotive software development. In *2022 International Conference on Embedded Software (EMSOFT'22)*. IEEE, 21–23.

[16] Kai-Björn Gemlau, Nora Sperling, and Rolf Ernst. 2022. Efficient timing isolation for mixed-criticality communication stacks in performance architectures. In *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'22)*. BEST PAPER AWARD.

[17] Mathieu Grenier, Lionel Havet, and Nicolas Navet. 2008. Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software (ERTS'08)*.
[18] ISO. 2018. 26262: Road vehicles-Functional safety. (2018).
[19] Christoph M. Kirsch and Ana Sokolova. 2012. The logical execution time paradigm. *Advances in Real-Time Systems* (2012), 103–120.
[20] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer.
[21] Nicolas Navet, Jorn Migge, Josetxo Villanueva, and Marc Boyer. 2018. Pre-shaping bursty transmissions under IEEE802. 1Q as a simple and efficient QoS mechanism. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems* 11, 2018-01-0756 (2018), 197–204.
[22] Institute of Computer and Network Engineering. 2020. IDA TSN Simulator. https://github.com/IDA-TUBS/IDATSNSimulator
[23] J. C. Palencia and M. Gonzalez Harbour. 1998. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*. 26–37. https://doi.org/10.1109/REAL.1998.739728
[24] J. C. Palencia, M. G. Harbour, J. J. Gutiérrez, and J. M. Rivas. 2017. Response-time analysis in hierarchically-scheduled time-partitioned distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 28, 7 (2017), 2017–2030. https://doi.org/10.1109/TPDS.2016.2642960
[25] D. Pannell, L. Chen, J. Dorr, W. Lo, M. Potts, H. Zinner, and A. Zu. 2019. P802.1DG – TSN profile for automotive in-vehicle ethernet communications. *Accessed: Jun* 29 (2019), 2022.
[26] D. Pannell, L. Chen, J. Dorr, W. Lo, M. Potts, H. Zinner, and A. Zu. 2019. Use cases-IEEE P802. 1DG V0. 4. *Accessed: Jun* 29 (2019), 2022.
[27] Paul J. Prisaznuk. 2008. ARINC 653 role in integrated modular avionics (IMA). In *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. IEEE, 1–E.
[28] Ola Redell and M. Torngren. 2002. Calculating exact worst case response times for static priority scheduled tasks with offsets and jitter. In *Proceedings. Eighth IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 164–172.
[29] Simon Schliecker, Mircea Negrean, Gabriela Nicolescu, Pierre Paulin, and Rolf Ernst. 2008. Reliable performance analysis of a multicore multithreaded system-on-chip. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. 161–166.
[30] Simon Schliecker, Jonas Rox, Matthias Ivers, and Rolf Ernst. 2008. Providing accurate event models for the analysis of heterogeneous multiprocessor systems. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. 185–190.
[31] Daniel Thiele, Philip Axer, and Rolf Ernst. 2015. Improving formal timing analysis of switched ethernet by exploiting FIFO scheduling. In *Proceedings of the 52nd Annual Design Automation Conference* (San Francisco, California) *(DAC'15)*. Association for Computing Machinery, New York, NY, USA, Article 41, 6 pages. https://doi.org/10.1145/2744769.2744854
[32] Daniel Thiele and Rolf Ernst. 2016. Formal worst-case performance analysis of time-sensitive ethernet with frame preemption. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA'16)*. IEEE, 1–9.
[33] Ken Tindell. 1994. *Adding Time-offsets to Schedulability Analysis*. Citeseer.
[34] Shane Tuohy, Martin Glavin, Ciarán Hughes, Edward Jones, Mohan Trivedi, and Liam Kilmartin. 2015. Intra-vehicle networks: A review. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 534–545. https://doi.org/10.1109/TITS.2014.2320605
[35] Guoqi Xie, Yanwen Li, Yunbo Han, Yong Xie, Gang Zeng, and Renfa Li. 2020. Recent advances and future trends for automotive functional safety design methodologies. *IEEE Transactions on Industrial Informatics* 16, 9 (2020), 5629–5642. https://doi.org/10.1109/TII.2020.2978889

Received 23 March 2023; revised 2 June 2023; accepted 13 July 2023