Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Supporting end-to-end data propagation delay analysis for TSN-based distributed vehicular embedded systems

Bahar Houtan *, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, Saad Mubeen

Mälardalen University, Sweden

ARTICLE INFO

Keywords:
Distributed embedded systems
Real-time systems
Time-sensitive networking

ABSTRACT

In this paper, we identify that the existing end-to-end data propagation delay analysis for distributed embedded systems can calculate pessimistic (over-estimated) analysis results when the nodes are synchronized. This is particularly the case of the Scheduled Traffic (ST) class in Time-sensitive Networking (TSN), which is scheduled offline according to the IEEE 802.1Qbv standard and the nodes are synchronized according to the IEEE 802.1AS standard. We present a comprehensive system model for distributed embedded systems that incorporates all of the above mentioned aspect as well as all traffic classes in TSN. We extend the analysis to support both synchronization and non-synchronization among the ECUs as well as offline schedules on the networks. The extended analysis can now be used to analyze all traffic classes in TSN when the nodes are synchronized without introducing any pessimism in the analysis results. We evaluate the proposed model and the extended analysis on a vehicular industrial use case.

1. Introduction

Vehicular distributed embedded systems are often modeled with chains of tasks and messages that can be distributed over two or more Electronic Control Unit (ECUs)[1] connected by a real-time network [1]. Traditionally, the in-vehicle communication was based on low-bandwidth and low-latency networks like Controller Area Network (CAN) [2]. Since CAN is an event-triggered communication protocol, the ECUs connected to the network are not synchronized. The traditional in-vehicle networks are unable to support high-bandwidth requirements in many complex vehicular distributed systems, in which realization of a higher level of autonomy of driving is envisioned. Realizing these vehicular systems is conditioned to incorporating a spectrum of functionalities, that range from handling high data-rate sensor readings, to gathering information of the vehicle's environment, and providing predictable responses to the corresponding inputs [3]. The communication standards utilized in future vehicular systems need to be flexible to allow accommodation of the new functions to the system over time. Recently, the IEEE Time-sensitive Networking (TSN) task group[2] developed a set of TSN standards that have emerged as a promising solution to support high-bandwidth and low-latency in-vehicle communication [4].

In vehicular distributed embedded systems, the data in a chain of tasks and messages propagates from the input to the output of the chain. The input corresponds to the first task in the chain, e.g., the task reading a sensor signal. Whereas, the output corresponds to the last task in the chain, e.g., the task producing an actuation signal. Note that any two neighboring tasks within the chain communicate using over-writable and non-consuming buffers, also called registers. This means, the writer task can over-write the previous data in the buffer, whereas the data stays in the buffer after it is read by the reader task. If tasks in such a chain are activated by independent activation sources with different periodicity (e.g., different periodic clocks), the data can propagate through more than one path from the input to the output of the chain. This leads to different types of delays that the data can experience while traversing through the chain. These delays are called data-propagation delays or end-to-end delays. The developers of the systems are required to verify, at the design time, that the specified timing constraints are satisfied. This can be achieved by performing the end-to-end data-propagation delay analysis of these systems [5–8].

* Corresponding author.
E-mail addresses: bahar.houtan@mdu.se (B. Houtan), mohammad.ashjaei@mdu.se (M. Ashjaei), masoud.daneshtalab@mdu.se (M. Daneshtalab), mikael.sjodin@mdu.se (M. Sjödin), saad.mubeen@mdu.se (S. Mubeen).

[1] We use the terms ECU, end-station and node interchangeably to refer to a single-core compute unit.
[2] https://1.ieee802.org/tsn
[3] Automotive systems are a subset of vehicular systems that include cars, trucks, construction vehicles, loading vehicles, moving cranes, to mention a few.
[4] SymTA/S tool has been acquired by Luxoft (https://www.luxoft.com).

https://doi.org/10.1016/j.sysarc.2023.102911
Received 15 October 2022; Received in revised form 27 April 2023; Accepted 22 May 2023
Available online 1 June 2023

A joint effort from the automotive[3] industry and academia identified the significance of these delays in vehicular systems and provided their formal semantics [5,6,9]. Eventually, the timing constraints corresponding to these delays were included in the automotive domain-specific modeling language EAST-ADL [10] and the AUTomotive Open System ARchitecture (AUTOSAR) standard [9]. In order to verify these timing constraints, the research community in collaboration with the R&D of the automotive industry developed the end-to-end data-propagation delay analysis [5–8]. This analysis is already implemented in several tools that are used in the vehicle industry, e.g., SymTA/S[4] [11] and Rubus [12]. Since CAN is the most widely used onboard real-time network in the vehicular domain, the existing analysis incorporated the response-time analysis for CAN [13,14] within the end-to-end data-propagation delay analysis. Furthermore, as CAN is an event-triggered network communication protocol and it does not support synchronization of the connected ECUs, the data-path computation algorithm within the existing end-to-end data-propagation delay analysis did not account for synchronization of ECUs.

The existing end-to-end data-propagation delay analysis and its data-path computation algorithm also support the precursor of TSN, called the Ethernet Audio-Video Bridging (AVB), which includes some classes of TSN. This is because AVB also supports event-triggered traffic and does not consider synchronization of ECUs. In that case, the response-time analysis of AVB [15] was incorporated within the end-to-end data-propagation delay analysis [16]. However, the TSN standards support synchronization of ECUs according to the IEEE 802.1AS standard. In particular, the ECUs should be synchronized when the Scheduled Traffic (ST) class in TSN is used, according to the IEEE 802.1Qbv standard. The real-time traffic mapped to the ST class is transmitted according to a schedule that is created offline. In general, when the IEEE 802.1AS standard is used then the ECUs in the TSN network should be considered synchronized regardless of which TSN traffic class, ST, AVB and Best-Effort (BE), is used. We identify that when the data-path calculation algorithm in the existing end-to-end data-propagation delay analysis, i.e. [5], is applied to the case of synchronized ECUs, the analysis results can be pessimistic (over-estimated) because the algorithm does not consider synchronization among the ECUs.

1.1. Paper contributions

In this paper, we extend the data-path computation algorithm within the existing end-to-end data-propagation delay analysis to support both synchronization and non-synchronization among the ECUs. The extended algorithm supports all traffic classes in TSN. Using the extended algorithm, the existing end-to-end data-propagation delay analysis can now be used to analyze all traffic classes in TSN networks, where the ECUs may or may not be synchronized, without introducing any pessimism (over-estimation) in the analysis results.

The **main contributions** in the paper are as follows:

– We extend the data-path computation algorithm within the existing end-to-end data-propagation delay analysis to support all traffic classes in TSN networks when the ECUs are synchronized using the IEEE 802.1AS standard. Unlike the existing algorithm, the analysis results with the extended algorithm do not include any pessimism when the ECUs in the TSN networks are synchronized. The extended algorithm is backwards compatible to support the analysis of all non-scheduled traffic (non-ST) classes[5] (AVB or BE) when the ECUs are not synchronized in the TSN networks.

– We present a comprehensive system model for distributed embedded systems to support the extended algorithm, which incorporates all traffic classes in TSN. The model can express distributed task chains that can contain various types of traffic supported by TSN, including the ST, AVB, and BE traffic.

– We demonstrate the applicability of the presented model and analysis to a vehicular industrial use case. We also perform comparative evaluation of the extended analysis with the existing analysis by analyzing the use case with the two analyses. Furthermore, the presented model and analysis are evaluated by experiments to show the effect of various configurations of ST class, receiver periods, and synchronization of the sender and receiver ECUs on the end-to-end data propagation delays.

1.2. Paper layout

The rest of the paper is organized as follows. In Section 2, we provide background on TSN and related works on timing analysis of TSN networks. Section 3 describes the end-to-end data propagation delays and elaborates on the over-estimation of delays if the existing analysis is applied to the ST class in TSN. Section 4 presents the proposed system model for distributed embedded systems, and furthermore Section 5 presents the extension to the existing end-to-end data propagation delay analysis. Section 6 presents a vehicular application case study. We compare the results of the existing analysis with our proposed extended end-to-end data propagation delay analysis. Furthermore, we present the use-case results and an experimental study to show the effect of various parameters on the end-to-end data propagation delays. Finally, we discuss the results in Section 6.4, and in Section 7 we conclude the paper.

2. Background and related work

2.1. Time-sensitive networking (TSN)

TSN standards are recently developed by the TSN task group in IEEE standardization. This set of standards can be seen as a toolbox containing various features to improve the performance of communication in several applications, e.g., automation and automotive applications [1,17,18]. According to the IEEE 802.1Q-2022 standards, the traffic classes are categorized into three categories of ST, AVB, and BE traffic. Among several features, the TSN standards allow temporal isolation of the ST traffic that is transmitted according to an offline schedule via the Gate Control List (GCL) as shown in Fig. 1. GCL is part of the Time-aware Shaper (TAS) that can realize the temporal isolation using a set of gates that control the transmission of traffic on a port of a TSN interface or switch. The gates can stop the transmission of lower priority traffic in favor of the urgent ST traffic class which in turn guarantees low-jitter transmission for the ST traffic (also known as preemption). In addition, the TSN standards define a Credit-Based Shaper (CBS) mechanism that allows reservation of bandwidth over the network for a set of traffic classes, known as the AVB classes. AVB traffic includes several classes starting from A (high priority) and can be up to eight as the number of queues per port is eight. It is very common to use only classes A and B in analysis and examples, while it is possible to have eight AVB traffic classes in the standard. Queues assigned to AVB class undergo the CBS mechanism for transmission. According to the CBS mechanism, a credit is configured per class of traffic on each TSN port and the traffic associated to the class can only be transmitted when the credit for that class is zero or positive. If the credit is negative, the transmission is on hold until the credit replenishes with a constant rate, known as the *idleSlope*, to zero or positive. The credit decreases when the transmission is happening with a constant rate, known as the *sendSlope*, and the summation of both values is equal to the port rate. Moreover, TSN can support legacy traffic transmission that do not need any timing guarantees, which is known as the BE traffic class.

[5] It is required to use synchronization when the ST traffic class in TSN is used.

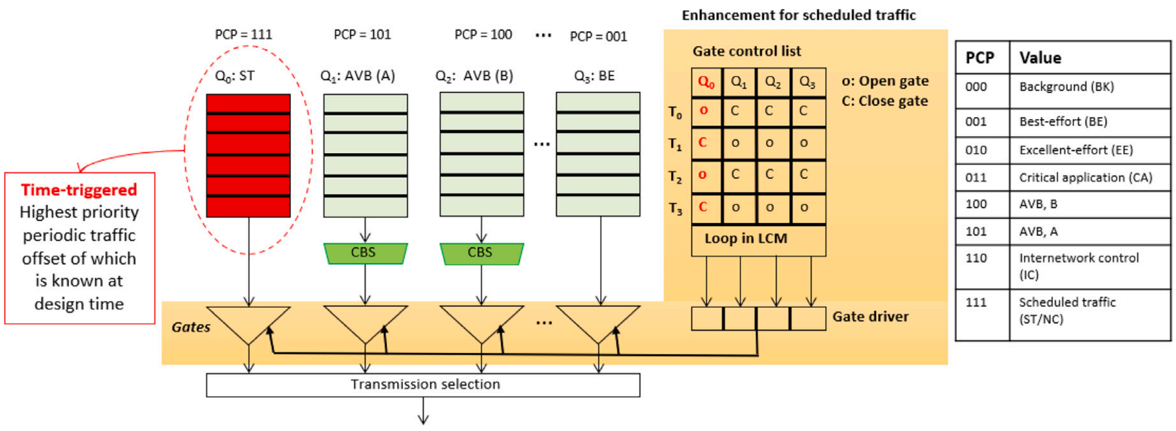


Fig. 1. TSN interface.

As shown in the example in Fig. 1, the eight traffic priorities in the Priority Code Point (PCP) table are mapped to the eight configurable queues at the egress port of the TSN switch. A queue can be configured to use each of the aforementioned mechanisms for passing the traffic with the associated PCP to the port. In this example, the PCP ”111” is configured to use the ST class, PCP ”101” and PCP ”100” are assigned to AVB class with the priorities A and B subsequently. The PCP ”001” is considered as BE traffic.

2.2. Related work

Several schedulability analysis techniques have been proposed in the literature to calculate the worst-case delays of traffic crossing through a TSN network. Among the techniques, many of them focused on the worst-case delays of the classes A and B frames under the CBS only, e.g., the work in [19] and the improved technique given in [20]. In addition, the work in [21] proposed a technique based on the trajectory approach to compute the delays of classes A and B. The technique obtains tighter bound of delays compared to the previous techniques, e.g., compared to the delays calculated by the approach in [19]. Later, the work in [22] proposed the notion of eligible interval that could provide a bound for delays per frame that leads to tighter analysis compared to the previous analysis techniques. The above-mentioned works solely consider the CBS in TSN networks. However, the TSN standards give various number of shapers and mechanisms that a network designer can select from. For instance, the proposal in [23,24] presented an analysis based on network calculus where it considers a TSN shaper called the Burst-Limiting Shaper (BLS).

Various schedulability analysis techniques focused on mechanisms other than the CBS, such as the gate mechanism. The analysis that is proposed in [25] computes the worst-case response times of classes A and B messages in TSN considering both CBS and the gate mechanism. However, the analysis considers a single-switch network, while industrial networks can consist of multiple switches. The work in [26] presented an analysis for calculating the accumulation of delays per link, whereas the works in [27,28] used network calculus to check the schedulability of TSN messages. In addition, the technique in [29] presented a response-time analysis for classes A and B messages considering the CBS and support for the ST that was earlier proposed in [30]. Furthermore, the work in [31] extends the technique in [29] for supporting the BE traffic in the response-time analysis. The traffic forwarding and shaping model in the latter work was different than the TSN standard models, as it was proposed before finalization of the first TSN draft.

A TSN network may also benefit from the preemption support along with the CBS and gate mechanisms according to the IEEE Time-sensitive Networking (TSN) task group.[6] Therefore, the work in [32] proposed an analysis considering frame preemption under the IEEE 802.3br standard. Further, the work in [33] presented a technique that calculates the worst-case response times of frames for classes A and B when the CBS, gate mechanism and frame preemption are used in a TSN network. Similarly, the work in [34] proposed a response-time analysis with the mentioned TSN features in combination with various modes, such as enabling and disabling the hold and release mechanism. The hold and release mechanism is defined in the TSN standards to prevent any possible jitter for the ST traffic due to transmission of lower-priority classes A and B.

To verify the timing behavior of TSN-based distributed embedded systems, not only the response times of tasks in the nodes and messages in the TSN network should be taken into account, but also the end-to-end data propagation delays in the chains of tasks that include TSN messages should be considered.

According to the classification in reference [35], there are three approaches for calculating the worst-case end-to-end delays: (1) simulation-based method that obtains the maximum end-to-end delays from a set of selected scenarios. The simulation approach does not necessarily show the maximum end-to-end delay, because it might not cover all the possible assumptions for the worst-case scenario; (2) model-checking based on an exhaustive search can provide exact worst-case end-to-end delays even on large-scale networks. However, these methods have high time complexity; and (3) analytical approach that provides upper bounds on end-to-end delays with a certain pessimism. Our work lies in the group of analytical approaches in the above mentioned classification.

From different perspective, the reference [36] mentions active and passive approaches for the end-to-end data propagation delay analysis. Active approaches optimize the pattern of task releases in a chain to achieve optimal delays. Whereas, passive approaches study the worst-case assumptions for the end-to-end delays in distributed embedded systems to find upper bounds on the delays. This paper and the works in [5,7,8,36–40] are among works in the passive category. In the following paragraphs of this section, we present some of the recent works on the end-to-end data propagation delay analysis.

The existing end-to-end data propagation delay analysis that computes the end-to-end delays incorporates the response-time analysis of various legacy real-time networks, such as CAN [7] and legacy Ethernet [41].

The work in [42] targets data age delay in cause–effect chains within one execution node. The tasks are synchronized inside an end-station, and they are scheduled with offsets. The aim of the paper is to find priorities, offsets and to optimize the design mapping for tasks to minimize the data age delays in a chain of tasks. Similarly, the work in [43] aims at finding offsets for the chain of tasks to optimize the end-to-end delays. The works in [42,43] belong to the active end-to-end data propagation analysis. Moreover, the paper [44] studies

[6] https://1.ieee802.org/tsn

the dependencies between the task instances. Such dependencies can be specified at early development stages to guarantee data age delay constraint.

The work in [36] performs end-to-end timing analysis for the systems with locally synchronized periodic tasks in one end-stations. The end-stations only support non-synchronized communication, i.e., via CAN, or FlexRay. Similarly, the work in [45] considers globally non-synchronized communication among the end-stations, while the tasks within the end-stations are considered synchronized. The authors in [45] propose a computationally-efficient analysis compared to the analysis in [36]. Besides, the work in [45] achieves a higher upper-bound on the data age delay than the work in [36].

The focus of the aforementioned works are only on data age delay, whereas our analysis also includes the analysis of the reaction delay. The reference [5] is a seminal work in the literature that introduces a formal framework for defining end-to-end delays in the periodic and register-based systems. The task model in [5] is based on Bounded Execution Time (BET) task model. In BET task model, the communication between the tasks is implicit where a task reads data from the register at its beginning and writes to the register at its end of execution. Furthermore, the work in the reference [46] proposes an end-to-end data propagation delay analysis that includes sporadic tasks based on BET task model. The work in paper [37] builds on the Logical Execution Time (LET) paradigm by proposing a system-level LET-based communication model for distributed embedded systems. LET is an inter-task communication model that augments the read/write access times (input and output of the task) to the task's physical execution time model. System-level LET models the communication between tasks of different end-stations. In the work presented in [37], the end-stations have their own local clocks (timeline). The global timeline is approximated based on the local timeline of the sender and receiver end-stations with a bounded error. In our end-to-end data propagation delay analysis, we rely on the determinism promised for ST traffic, and exclude the global synchronization error for analyzing transactions that utilize ST traffic. The work in [47] considers globally non-synchronized and locally synchronized task chains and propose a method to calculate a limited number of timed-paths[7] that lead to the maximum end-to-end delays. We extend the timed-path approach based on the works in [5,7,8,41], which calculate all possible timed-paths but within a bounded window equal to twice the hyperperiod or the Least Common Multiple (LCM) of the periods of all involved tasks in the chain. Consequently, our algorithm finds the timed-path that leads to the worst-case end-to-end delays after considering all possible cases.

While the majority of the works focus on providing methods for calculating the maximum end-to-end delays in the chains, the work in [48] discusses robustness margins around the end-to-end timing constraints. The work in [48] employs BET and system-level LET communication model, and further studies the variations in the tasks' response times and the influence on the robustness of the system (i.e, variations in the end-to-end delays). In our evaluations, we also take into consideration the changes in the end-to-end timing delays by making variations in the configuration of ST traffic and the periods of the receiver tasks. We focus only on the BET and consider pre-defined end-to-end timing constraints for the system under evaluation.

The work in [38] proposes an end-to-end data propagation delay analysis for the chains of tasks in the context of Robot Operating Systems (ROS). The analysis prior to [38] mainly focused on analyzing periodic and sporadic tasks. The work in [38] extends the end-to-end timing analysis to support ROS2 task chains that deal with a mix of time-triggered and event-triggered functions. Besides, the network model in [38] is based on publisher–subscriber communication model. The communication model is therefor inherently non-synchronized. The proposed analysis in our work is only applicable to periodic tasks.

In comparison to the aforementioned works, this paper aims at extending the data-path calculation algorithm within the existing end-to-end data propagation delay analysis [5,7,8,41,47,49] to support analysis of all traffic classes in TSN where nodes can be synchronized or non-synchronized. These works have also been implemented in tools to support model- and component-based software development of vehicular embedded systems, e.g., [7,39,40], all of which are considering the BET task modeling paradigm.

The existing timed-path calculation used in the end-to-end data propagation delay analysis algorithms, such as [5,47], applies to traffic classes that do not require offline schedules, i.e., AVB and BE. For example, the work in [47] provides end-to-end analysis for the synchronized communication between sender and receiver end-stations, but it only considers the case when non-ST are transmitted between synchronized end-stations, i.e. AVB or BE.

To the extent of our knowledge, there are a few works that aim at realizing end-to-end data propagation delay analysis for interconnected end-stations in distributed embedded systems that are based on the system-level LET model, such as [36–38,48]. Unlike the previous works, this paper considers that the ST messages are scheduled with offsets (with globally synchronized end-stations).

These two models can be mapped to each other according to [48]. We chose to use the BET model because it is already integrated to several tools (including industrial tools) that support model- and component-based software development of vehicular embedded systems, e.g., in [39,40].

3. End-to-end data propagation delays

Embedded real-time systems are often modeled with chains of tasks and messages. To verify the timing behavior of these chains, not only their end-to-end response times need to be calculated and compared against the corresponding deadlines, but also the end-to-end data propagation delays (data age and reaction time) should be calculated and compared with the corresponding data age and reaction time constraints. The timing constraints on the data age and reaction delays are often specified on these distributed chains. The constraint on the data age delay is important, in particular, for control applications where freshness of the data is of value. Whereas, the reaction constraint is important in applications where the time of the first reaction to the input event is of value. These constraints are included in the timing model of the AUTOSAR standard [9] and are translated to several modeling languages in the vehicular domain [50].

3.1. Data propagation delays in single-node embedded systems

In order to explain the data age and reaction time delays, consider a task chain consisting of three tasks τ_1, τ_2 and τ_3, as shown in Fig. 2. All tasks belong to a single-core node and are activated independently. The periods of activation for tasks τ_1, τ_2 and τ_3 are 8 ms, 8 ms and 4 ms, respectively. The Worst-Case Execution-Time (WCET) of each task is assumed to be 1 ms. For simplicity, we assume that the priority of τ_1 is higher than the priority of τ_2 and the priority of τ_2 is higher than the priority of τ_3. By this priority assignment policy, we ensure that the precedent elements in the chain should be executed before their subsequent elements in the chain. The tasks use register-based communication, i.e., they communicate with each other and with their environment by means of writing data to/ and reading data from the registers. The registers are of non-consuming type. This means that data stays in the register after the reader has read the data. Furthermore, the registers are over-writable, i.e., if the writer is faster than the reader then the previous data in the register can be overwritten by the new data before the reader can read the previous data. The data read by τ_1

[7] We use the terms timed-paths and data-paths interchangeably to refer to the path that the data traverses from one task to another within the task chain.

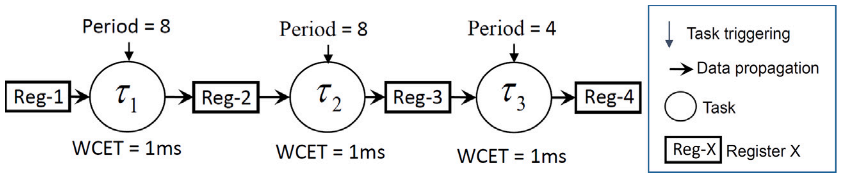


Fig. 2. An example of a task chain that uses register-based communication.

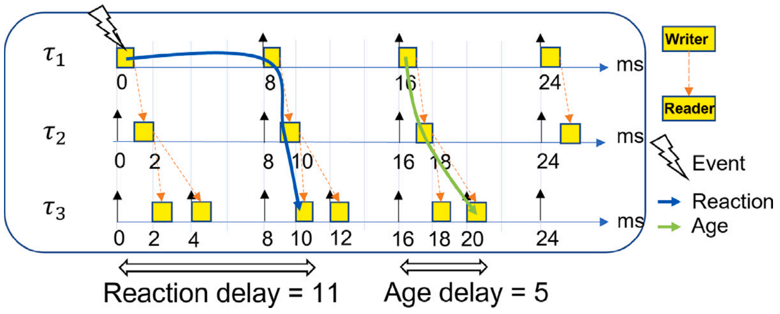


Fig. 3. Data age and reaction time delays in the task chain depicted in Fig. 2.

from Reg-1 corresponds to the input of the chain. Similarly, the data written to Reg-4 by τ_3 corresponds to the output of the chain.

As the tasks are activated independently and some tasks have different periods, the data traverses through the chain via multiple paths from the input to the output of the chain as shown in Fig. 3. These paths are called timed-paths (also referred to as data-paths). Due to multiple timed-paths, there can be various delays to deliver the data from the input to the output of the chain.

The data age delay is the time elapsed between the arrival of data at the input and the latest availability of the corresponding data at the output. In the data age delay analysis, we are interested in identifying the longest time difference between the input data and the last sample of corresponding output data. On the other hand, the reaction delay corresponds to the earliest availability of the data at the first instance of the output corresponding to the data that *just missed* the read access at the input. An event (corresponding to availability of data) is considered as readable by an instance of a task, if it occurs at or before the activation of the task. If the event happens just after the activation of the task instance, the data is not readable to this instance, i.e., the data is just missed by the current instance of the task. The missed data is read by the next instance of the task. This is illustrated by the white thunderbolt in Fig. 3, where the first instance of τ_1 at time 0 misses the data but the same data is read by the next instance of τ_1 at time 8.

Possible data age and reaction delays in the chain in Fig. 2 are shown in Fig. 3. On the one hand, the data from the event happening a bit before time 16 is accessible to the third instance of τ_1 (activated at time 16). In such a case, the latest impact of this event is available at the output of the chain until 5 ms after the occurrence of the event (data age delay). On the other hand, the sampling of the data coming from the event happening a bit after the time 0 is delayed until the time 8, where the data can be read by the second instance of τ_1. Accordingly, the earliest time the impact of the data appears at the output of the chain is 11 ms after the occurrence of the event (reaction time delay).

3.2. Data propagation delays in distributed embedded systems

The data propagation delays are equally valid in distributed embedded systems. Let us consider a distributed task chain in a distributed embedded system depicted in Fig. 4, where two nodes are connected via a network. In this example, the tasks are activated periodically with periods of 6 ms and 3 ms, respectively. Task τ_1 in Node 1 sends a message to task τ_2 in Node 2 through the network.

Depending on the type of network, we may have different possible timed-paths through which the data can propagate from the sender task to the receiver task. For example, when the network is not capable of initiating communication independent of the sending tasks, a message can only be queued for transmission at the network interface by the sending task. This is the case of many event-triggered network protocols, like CAN [2]. In this case, the message inherits its period from the sender task. Furthermore, the timed-paths in a distributed task chain also depends upon whether the network supports synchronization of nodes. For example, TSN supports synchronization among the end stations via the IEEE 802.1AS standard, whereas the CAN protocol does not support synchronization. Fig. 5 shows an execution trace when the nodes are synchronized in the system that is shown in Fig. 4. The data age and reaction time delays in this distributed chain are identified as 7 ms and 10 ms, respectively.

A possible execution trace of the distributed task chain in Fig. 4 when the nodes are not synchronized is shown in Fig. 6(a). To create worst-case conditions when the nodes are not synchronized, we assume that the receiver task τ_2 is activated “just before” the arrival of the message at the receiver node. Hence, the current instance of τ_2 (the first period activation) will miss the read access of the message. The message will be read by the next instance of τ_2 (second period activation) as shown in Fig. 6(a). The corresponding data age delay is identified as 9 ms as shown in Fig. 6(a).

To increase readability, we draw the same execution trace separately for the case of reaction time delay in the distributed task chain (as shown in Fig. 4) when the nodes are not synchronized as depicted in Fig. 6(b). In Fig. 6(b), the first instance of τ_1 is activating the first instance of the message m_1. According to the assumption for the reaction delay, the first instance of τ_1 has missed the sampling of the chain’s input event, thus the first instance of m_1 does not deliver valid data from the input event to the receiver task. However, as the second instance of τ_1 reads the input event, the second instance of the message m_1 also holds fresh data. Since τ_2 is not synchronized with τ_1, the worst-case assumption is that τ_2 is activated a small amount of time earlier than arrival of the message that holds the sampled data. Consequently, the first instance of τ_2 misses to read data of the event, which is being written by the second instance of m_1. But, at the next instance of τ_2 (second period activation), τ_2 is able to read the incoming data from m_1. Accordingly, the reaction delay is 12 ms.

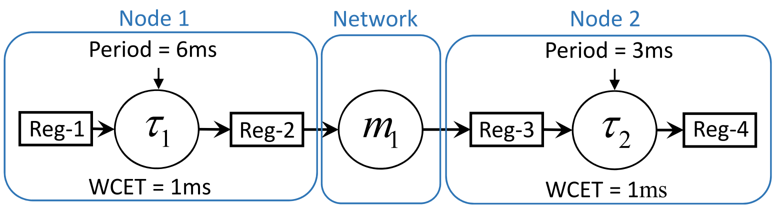


Fig. 4. A multi-rate chain in a distributed embedded system.

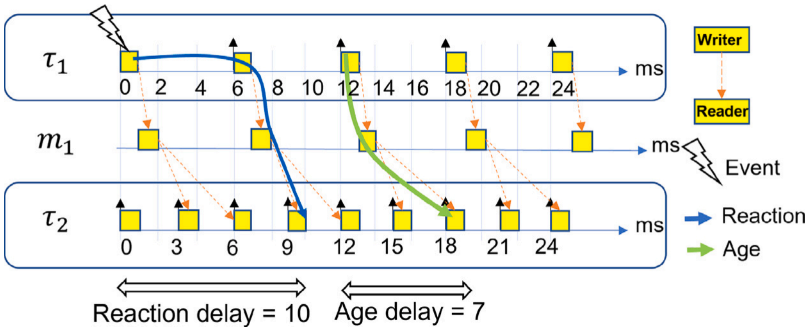
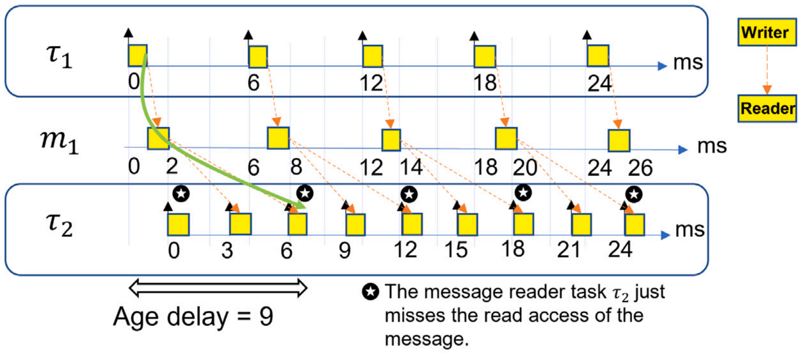
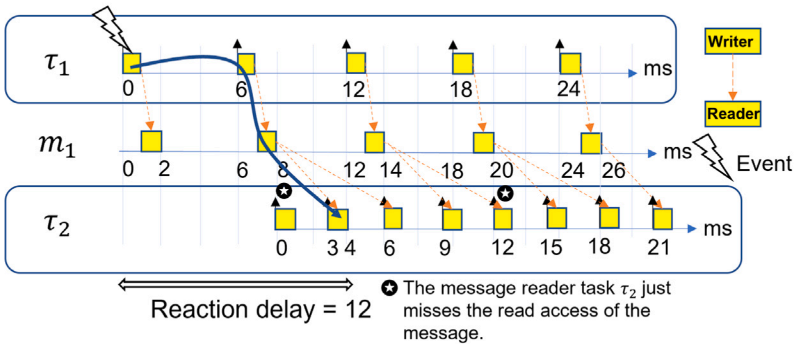


Fig. 5. A possible execution trace for the distributed embedded system example shown in Fig. 4 when source and destination end-stations are synchronized.


τ_1
m_1
τ_2
ms
Writer
Reader
Age delay = 9
The message reader task τ_2 just misses the read access of the message.

(a) Data age delay.

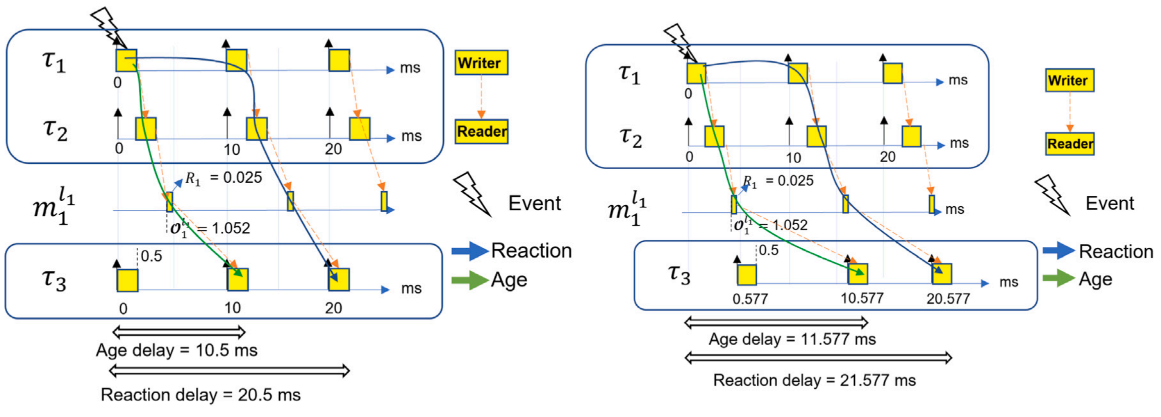

τ_1
m_1
τ_2
ms
Writer
Reader
Event
Reaction delay = 12
The message reader task τ_2 just misses the read access of the message.

(b) Reaction time delay.

Fig. 6. A possible execution trace for the distributed embedded system example shown in Fig. 4 when source and destination end-stations are not synchronized.

3.3. Need for extending the timed-path calculation algorithm

The timed-path computation algorithm in the existing end-to-end data propagation delay analysis implicitly assumes that the nodes are not synchronized. Hence, the worst-case assumption for the message receiving task is that the task is released for execution "just before" the arrival of the message. This implies that the current instance of the task cannot read the message and hence the next instance of the task will read the message. This may not be true in the case of TSN when the nodes are synchronized using the IEEE 802.1AS standard. If the existing algorithm is applied to synchronized nodes, as is the case of TSN, then the calculated delays can be pessimistic (i.e., over-estimated). This can be seen in the figures Figs. 5, 6(a), and 6(b), where data age and reaction time delays of 7 ms and 10 ms are calculated for synchronized end-stations (Fig. 5). In comparison to the case of non-synchronized end-stations respectively (Figs. 6(a), and 6(b)), data age and reaction delays of 9 ms and 12 ms are calculated respectively.



(a) τ_1 and τ_2 are synchronized. (b) τ_1 and τ_2 are not synchronized.

Fig. 7. End-to-end delays for an ST message.

Another example can be seen in Fig. 7, where the existing end-to-end data propagation delay analysis is performed on a transaction between two end-stations that use ST class for transmission of message m_1 via link l_1[8]. In the sender end-station, the periods of each of tasks τ_1 and τ_2 are 10 ms. The message m_1 is sent by τ_2 in the sender end-station. The period of the receiver task (τ_3) is 10 ms, and it reads the data coming from the message m_1 from the link l_1. The offset of the message is set to 1.052 ms, and the message is expected to finish its transmission at 0.025 ms later than the offset of the message at l_1. The WCET of each task is 0.5 ms. In the ideal condition for ST traffic when the sender and receiver end-stations are synchronized as shown in Fig. 7(a), the data age and reaction time delays are subsequently 10.5 ms, and 20.5 ms. As seen in Fig. 7(b), the data age and reaction time delays for this transaction in the case of non-synchronized nodes are pessimistic (over-estimated), i.e., 11.577 ms and 21.577 ms respectively. This is not desirable, since ST traffic is most commonly applied in critical applications that require precise calculation of the delays. Therefore, the data-path computation algorithm in the existing analysis requires extension to support synchronized nodes without introducing any pessimism in the analysis results.

4. System model

In this section, we formally present the system model of a distributed embedded system that consists of two or more end-stations (single-core nodes, compute units, or ECUs) that are connected by a TSN network. The system $\mathcal{S}$ consists of a set of *transactions*, denoted by Γ, a set of *end-stations*, denoted by $\mathcal{E}$, and a *network*, denoted by $\mathcal{N}$. The system is formally expressed by the following tuple.

$$\mathcal{S} := \left\langle \Gamma, \mathcal{N}, \mathcal{E} \right\rangle \tag{1}$$

A transaction (denoted by Γ) represents the model of a distributed task chain that consists of two or more tasks. The chain of tasks either can be executed within one end-station; or tasks of different end-stations can communicate with each other via one or more messages in the network ($\mathcal{N}$). Multiple transactions can exist in the system model. The set of transactions are formally expressed subsequently by Eq. (2):

$$\Gamma := \{\Gamma_1, \dots, \Gamma_{|\Gamma|}\} \tag{2}$$

The set of end-stations in the system are represented by Eq. (3).

$$\mathcal{E} := \{\mathcal{E}_1, \dots, \mathcal{E}_{|\mathcal{E}|}\} \tag{3}$$

[8] A detailed system model with all notations will be described in Section 4.

4.1. End-station model

An end-station $\mathcal{E}_i$ may consist of one or more tasks as shown in Eq. (4):

$$\mathcal{E}_i := \{\tau_{ij1}, \dots, \tau_{ijk}\} \tag{4}$$

where i is the index of the end-station to which the task belongs. Note that a task in an end-station may be a part of one or more transactions. Hence, j represents the transaction index. Finally, k represents the unique identifier of the task within the scope of the end-station.

4.2. Task model

The properties of a task are specified by the tuple in Eq. (5).

$$\tau_{ijk} := \left\langle P_{ijk}, C_{ijk}, T_{ijk}, J_{ijk}, O_{ijk} \right\rangle \tag{5}$$

where, C_{ijk} is the task's WCET, T_{ijk} is the task's period, P_{ijk} is the task's priority, and J_{ijk} is the task's release jitter. Moreover, O_{ijk} represents the offset of the task.

Some other properties of the task are calculated using the aforementioned information in the task's tuple. Firstly, the activation time of the nth instance of the task τ_{ijk} (denoted by $\alpha_{ijk}(n)$) can be obtained using the task's period and offset based on Eq. (6). Moreover the worst-case response time of the task is indicated by R_{ijk}. Also, the nth instance of the task τ_{ijk} is denoted by $\tau_{ijk}(n)$.

$$\alpha_{ijk}(n) = n * T_{ijk} + O_{ijk} \tag{6}$$

4.3. Network model

The network attributes are indicated by a set of parameters in Eq. (7):

$$\mathcal{N} := \langle s, \mathcal{L}, \mathcal{I} \rangle \tag{7}$$

where s is the overall network speed. We assume the network operates with the same speed on all of the links. $\mathcal{L}$ holds the set of links in the network. We consider that each link creates a bi-directional connection between an end-station and a switch or between two switches. All switches in the network are TSN switches, hence there can be different traffic classes in the network. We indicate the traffic classes by the set $\mathcal{I}$ in Eq. (8), where AVB can be classes A, B or other classes that undergo the CBS. Moreover, ST and BE represent the scheduled traffic and best-effort traffic classes respectively.

$$\mathcal{I} = \{AVB, ST, BE\} \tag{8}$$

A message in the network is indicated by m_{jk}, where the subscript j identifies the transaction to which the message belongs. Furthermore, the subscript k is the unique identifier of a message within the scope of the network. Eq. (9) shows the set of attributes defining the properties of a message.

$$m_{jk} := \langle P_{jk}, Size_{jk}, T_{jk}, J_{jk}, \mathcal{L}_{jk}, \mathcal{O}_{jk} \rangle \tag{9}$$

where the priority of the message is denoted by P_{jk} that specifies the TSN class from which the message is transmitted, e.g., class ST, AVB (A, B, or etc.) and BE. In this model, the ST class has the highest priority, while AVB has a lower priority than ST. Moreover, the BE class has the lowest priority among all the other classes. The size of data (payload size in bytes) is indicated by $Size_{jk}$. We assume that all messages are periodic, therefore T_{jk} is the period of the message. The release jitter of the message is indicated by J_{jk}. The set of links assigned as the route of the message from the source end-station to the destination end-station is stored in the set $\mathcal{L}_{jk}$. Furthermore, the set of offsets of the message at each of the links specified in the set $\mathcal{L}_{jk}$ is stored in the set $\mathcal{O}_{jk}$. We assume that we only know the offset of the ST traffic as it is scheduled offline. Therefore, the set $\mathcal{O}_{jk}$ for non-ST traffic (AVB or BE) is assumed to be empty, i.e., $\mathcal{O}_{jk} = \{\}$.

Based on the aforementioned properties, we can calculate other properties of the message m_{jk}, such as the transmission time (C_{jk}) as well as the worst-case response time (R_{jk}) by utilizing the response-time analysis of various classes in TSN [31]. Moreover, the activation time of the nth instance of the message m_{jk} at link l is calculated by Eq. (10), where, O^l_{jk} is the offset of m_{jk} on link l.

$$\alpha^l_{jk}(n) = n * T_{ijk} + O^l_{jk} \tag{10}$$

We assume that both ST and non-ST traffic inherit the period from the corresponding sending task. Therefore, T_{ijk} indicates the period of the kth task (sending task) belonging to the ith end-station and being part of the jth transaction. Moreover, we denote the nth instance of the message m_{jk} by $m_{jk}(n)$.

4.4. Transaction model

A transaction Γ_j represents the model of a distributed task chain that consists of two or more tasks that communicate with each other via one or more messages. The data read by the first task of the transaction is considered as the input of the transaction, and the data written by the last task of the chain corresponds to output of the transaction. The period of the transaction is denoted by T_j. Note that this model limits the number of message to one per transaction.

Fig. 8 shows an example of a TSN-based distributed embedded system with the presented model. There are two end-stations that are connected to one TSN network. More specifically, end-station $\mathcal{E}_1$ and $\mathcal{E}_2$ are connected via links l_1 and l_2 to switch 1 ($SW1$). There are two transactions in the system, namely Γ_1 and Γ_2, as shown in Fig. 8. These transactions are further elaborated in Fig. 9.

According to the example in Fig. 8, Γ_1 is a transaction that is within a single end-station ($\mathcal{E}_2$) and only includes tasks from $\mathcal{E}_2$. Hence, $\mathcal{E}_2$ is both initiator and terminator end-station of Γ_1. As shown in Fig. 9, transaction Γ_1 initiates and terminates inside the end-station $\mathcal{E}_1$. On the other hand, transaction Γ_2 is distributed over two end-stations. Γ_2 initiates from $\mathcal{E}_1$ and terminates in $\mathcal{E}_2$. The message $m_{2,1}$ travels from link l_1 and l_2 between the transaction initiator and terminator. It should be noted that, the transaction model does not consider forking and joining of tasks in a transaction.

4.4.1. Trigger modes

According to [51], the assumptions on the activation times of the tasks and messages in a distributed transaction affect the delays of the transaction. In this paper, we assume that each entity in a transaction, regardless of being a task or message, can be triggered in two modes,

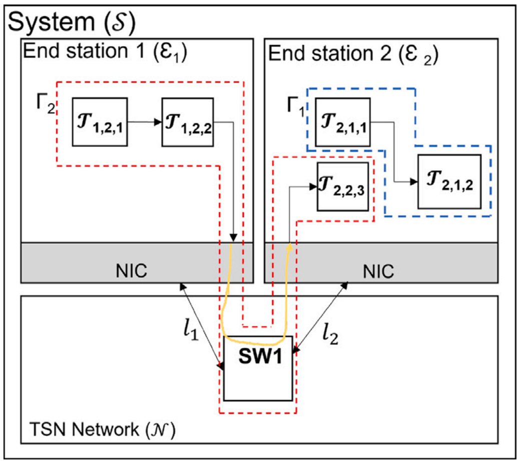


Fig. 8. Example of a distributed vehicular embedded system based on TSN.

i.e. "$Dependent$" or "$Independent$". The trigger mode is selected by the parameter $triggerMode$, as shown in Eq. (11).

$$triggerMode := \{Dependent, Independent\} \tag{11}$$

A task can be independently triggered by an event source, e.g., a periodic clock. Additionally, a task can be triggered based on (i) an activation signal from a predecessor task, (ii) receiving data from a predecessor task, or (iii) a combination of both.

4.4.2. Message activation time

If the message is ST, it is triggered independently. This means that the message is triggered based on static offsets defined for it at each of the links specified in its route to the destination end-station. In case of the dependent trigger mode, the message is triggered at the end of the execution of its sender task. For example, the message assigned to non-ST classes in TSN networks (AVB or BE) can use the link as soon as the message's sender task completes its execution as well as the bandwidth on the link is available, and there are no higher priority messages that need to be transmitted.

4.4.3. Receiver task's activation time

Activation time of the message receiving task can be calculated based on different assumptions. For instance, if the sender and receiver end-stations are synchronized, the first instances of their tasks are assumed to be activated simultaneously. In such a case, the receiver task polls for the TSN network to read data from the messages. Accordingly, the activation time of the receiving task within a synchronized network is calculated by the generic equation in Eq. (6). Where each instance n of the task within the receiving end-station is released periodically (T_{ijk}) and can have offset specified by the parameter O_{ijk}. In the non-synchronized network, the existing analysis assumes that the receiver task gets read access as soon as a message is available on the network. Hence, the activation time of the message receiving task is represented by Eq. (12).

$$\alpha_{ijk}(n) = n * T_{ijk} + O_{ijk} + R_{jk} \tag{12}$$

4.4.4. Transaction constraints

A timing constraint on transaction, denoted by Cr_j, defines the maximum allowed value of the delays, such as data age (Age_j) and reaction time ($Reac_j$). Moreover, a deadline constraint (D_j) can be specified on a transaction. Deadline constraint corresponds to the end-to-end response time, i.e., response time of the transaction from its input to its output. These constraints are shown in Eq. (13) for transaction Γ_j:

$$Cr_j := \{Age_j, Reac_j, D_j\} \tag{13}$$

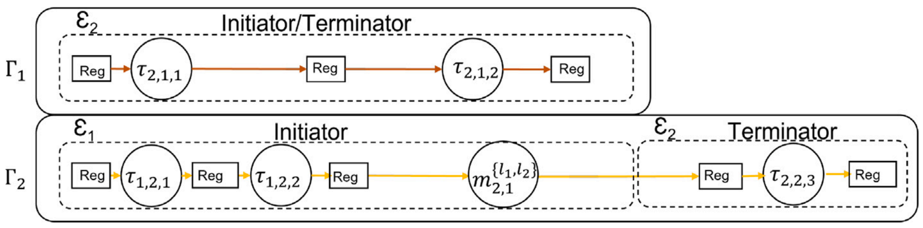


Fig. 9. Example of transactions.

5. End-to-end data-propagation delay analysis

This section presents the end-to-end data propagation delays analysis of task chains that are distributed over TSN network. The analysis is based on the existing analysis [5,7] that considers legacy networks like CAN. The end-to-end data propagation delay analysis requires computation of all relevant data-paths (also called reachable timed-paths) within the distributed task chains. In TSN unlike CAN, different traffic classes are to be analyzed in the same network, which would require incorporating synchronized and non-synchronized end-stations. The data-path computation algorithm in the existing end-to-end data propagation delay analysis [5,7] only support non-synchronized end-stations. In this section, we propose an extended algorithm for TSN networks which covers all the TSN traffic characteristics while supporting both synchronized and non-synchronized end-stations.

5.1. Reachable timed-paths

The order of read and write by each instance of the tasks from the input to the output of the transaction is represented by a set of timed-paths. These timed-paths track the propagation of data from the input to the output of the transaction. Therefore, each transaction can have a set of timed-paths. A timed-path belonging to the transaction Γ_j is denoted by tp_j^i, where i is the ID of the timed-path. A valid timed-path between a writer and reader task instance (the term "instance" is equivalent to the term "task job") is selected according to a set of conditions as presented in [5]. In the following paragraphs of this section, we briefly explain the Boolean functions for checking the reachability of the timed-paths as presented in [5].

The first condition in identifying a valid timed-path checks whether the reader task (say τ_{dbe}) is activated after the activation of the current instance of the writer task (say τ_{abc}). Violating this condition is also known as activation time-travel ($att()$), therefore Eq. (14) defines this condition in negated form. Because, the activation time-travel should not happen in valid timed-paths. Note that $\alpha(w)$ shows the activation time of the wth instance of the task computed according to Eq. (6).

$$att(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) = \alpha_{dbe}(r) < \alpha_{abc}(w) \tag{14}$$

Eq. (14) is true in case the activation time-travel happens between the writer and reader tasks. Otherwise, for valid timed-paths it is desirable when the condition in Eq. (14) is false.

Moreover, the completion of the writer and reader tasks should not overlap in a valid timed-path. Therefore, the next reachability condition checks whether the activation time of the reader task is after the completion of the current instance of the writer task (according to R_{abc}, the worst-case response time of the writer task, referring to the system model presented in Section 4). Violating this condition is also known as critical condition and is represented by the critical function ($crit()$), as shown in Eq. (15). $crit()$ function returns true if the reader task τ_{abc} is activated before the writer task is completed. In such a case, the reader task misses the data from the writer task. Otherwise, the function in Eq. (15) returns false, which is desirable for valid timed-paths.

$$crit(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) = \alpha_{dbe}(r) < \alpha_{abc}(w) + R_{abc} \tag{15}$$

The instance number of each instance of writer and reader tasks are indicated in Figs. 10(a) and 10(b). Fig. 10(a) shows an example of the case where the reader task is activated just after the writer task is completed. In this case, we have a reachable timed-path from the first instance of the writer task to the first instance of the reader task. Note that the timed-path from the second instance of the writer task to the reader task's first instance is not reachable, and it will be excluded by Eqs. (14) and (15).

Moreover, the reader and writer task instances can only overlap when they are activated at the same time in the same single-core end-station. In such a case, the reader task does not miss the writer task's data, if the priority of the reader task (P_{dbe}) is lower than the priority of the writer task (P_{abc}) as shown in Eq. (16). If the reader task executes before the writer task, then the reader task needs to wait until its next period activation in order to read the fresh data from the writer task. This is taken into account according to the wait function ($wait()$) in Eq. (16), where P indicates the priority of the task according to the system model in Section 4.1.

$$wait(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) = P_{dbe} < P_{abc} \tag{16}$$

Fig. 10(b) shows an example of a timed-path between two tasks inside an end-station. If both writer and the reader tasks are activated at the same time, there is a reachable timed-path between the writer and the reader task, provided that the writer task has executed before the reader task.

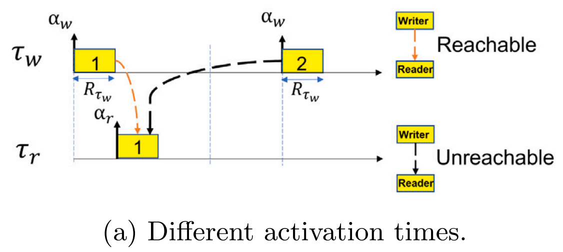
Accordingly, if the writer and reader task instances are from two different end-stations, the reachability of the timed-path through the network is obtained referring the worst-case response time of the message communicated between the writer and reader tasks, and the activation time of the writer task instance, as shown in Fig. 11. For instance, if there is a writer task τ_w and a reader task τ_r which communicate by a message Msg, there are three different timed-paths as shown in Fig. 11. However, only one of those timed-paths is a reachable timed-path from the input to the output of the transaction.

The forward reachability of two tasks in the timed-path, according to the aforementioned functions ($att()$, $crit()$ and $wait()$), is examined based on the forward reachability function $forw()$ as presented in Eq. (17).

$$\begin{aligned} &forw(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) = \\ &\neg att(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) \wedge \\ &(\neg crit(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) \vee wait(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r))) \end{aligned} \tag{17}$$

Further, it is important to note that Eq. (17) does not cover all the cases to check the reachability of the timed-paths, because it can happen that two instances of a writer task reach to an instance of a reader task, e.g., when the period of the writer task is shorter than the reader task. In this case, the data that is accessible to the reader task will be overwritten by subsequent writer task instances. We make sure that only the last writer task instance reaches the reader task instance, i.e., there is no next writer task instance that reaches this reader task instance. This can be detected when the function in Eq. (18) returns true, where $\tau_{abc}(w+1)$ represents the next instance of the task instance $\tau_{abc}(w)$.

$$\begin{aligned} &reach(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) = forw(\tau_{abc}(w) \longrightarrow \tau_{dbe}(r)) \\ &\wedge \neg forw(\tau_{abc}(w+1) \longrightarrow \tau_{dbe}(r)) \end{aligned} \tag{18}$$



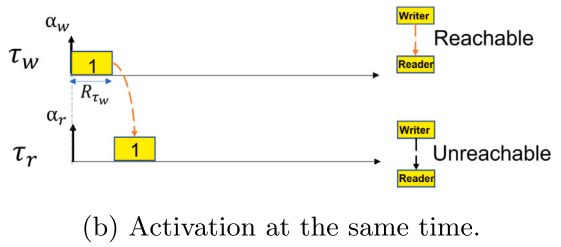


Fig. 10. Reachability conditions for tasks in the same end-station.

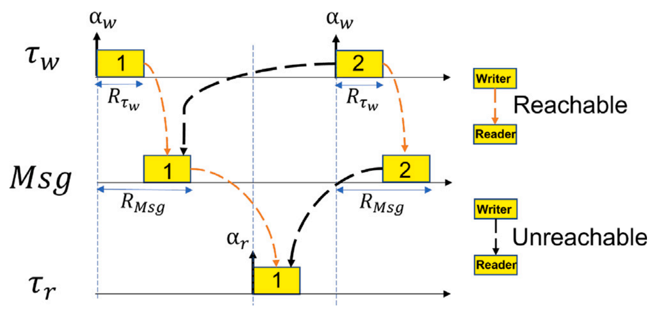


Fig. 11. Timed-paths.

After checking the reachability between two task instances, we check for the whole timed-path by evaluating every two consecutive task instances in the timed-path from the first task until the last task. We show this by Eq. (19) where a timed-path tp_j^i belongs to the transaction Γ_j. For simplicity of the equation, two consecutive task instances in the timed-path are shown by τ_w and τ_r.

$$reach(tp_j^i) = \prod reach(\tau_w \longrightarrow \tau_r) \tag{19}$$

We finally evaluate all possible timed-paths in the transaction Γ_j to obtain all reachable (valid) timed-paths in a set TP_j^{reach}, according to Eq. (20) assuming that there are z timed-paths in the transaction.

$$TP_j^{reach} = \{reach(tp_j^i); i = 1..z\} \tag{20}$$

5.2. Accounting for the ST messages

Each ST message has a deterministic schedule at each link within its route from the sender (writer) task to the receiver (reader) task. The activation time of an ST message on its last link (the time it is available to the reader task instance) is not directly dependent on the response time of the sender task or the response time of the message itself on the previous links. Although, some parameters such as the response time of the sender task, the transmission time of the ST message per link and/or other optimization objectives can be accounted for when defining the offset of the ST message per link. This is opposed to the case of a non-ST message, where the activation time of the message on the last link between the sender and receiver tasks is dependent upon both the response time of the sender task and its own response time on the previous links. Whereas, the ST messages are isolated by the time slots, which are configured offline on each link in their route to the receiver tasks.

ST offsets at the subsequent links in the route of the message are defined in a way to satisfy a set of scheduling constraints as presented in our previous work [52]. The scheduling constraints differ from the timing constraints which were mentioned earlier in this paper. The scheduling constraints are a set of logical rules set on the message's set of offsets per link to find satisfiable values of offsets which lead to a feasible offline schedule. The TSN schedules are configured offline, and before performing the data propagation delay analysis.

The most important set of scheduling constraint are as follows: (1) constraint on the frame size; (2) constraint on overlapping of messages on a link; (3) constraint on the order of traversed links; and (4) constraint on deadline satisfaction. Firstly, the constraint on the frame ensures that the value of the offset on one link does not force the arrival of the message after its next period activation. Secondly, the overlapping constraint checks whether the offset of two different messages on the same link will not cause overlapping time slots for these two messages. Thirdly, the order of the offset for the same message on subsequent links must consider the propagation of the message from the source end-station's link to the sink end-station's link (avoiding time-travel). Fourthly, the offsets per links in the route of the message must enforce the message to arrive to the destination end-station before the message's deadline (implicitly the next period activation of the sending task).

According to the feasible TSN schedule, the reachability of an instance of an ST message to the instances of the reader task is determined considering the offsets per link along the route of the message. In this case, the existing constraint to find reachable timed-path requires to trace through all the hops in the route of the ST message and take into account the activation times of the ST message per link.

An example of a transaction is shown in Fig. 12, in which the ST class is utilized for the communication between two end-stations. The two end-stations are connected via the links l_1 and l_2, therefore the instance of the ST message at link l_1 is scheduled by the offset $O_{1,1}^{l_1}$ and it is notated by $m_{1,1}^{l_1}$. Likewise, the instance of the ST message on the link l_2 is indicated by the notation $m_{1,1}^{l_2}$ and its offset on the link is defined by $O_{1,1}^{l_2}$. Moreover, the reachable timed-paths in the transaction (Γ_1) shown in this example are subsequently, tp_1^1, tp_1^2, tp_1^3 and tp_1^4. For instance, tp_1^1 starts with the first instances of the first task ($\tau_{1,1,1}$) and the second task ($\tau_{1,1,2}$) of the source end-station. Then the message uses the bandwidth of the links in its path according to its offset at each of the links.

As it can be seen in Fig. 12, the analysis has to be extended to support multiple activations of the message on several links as the ST messages have deterministic activation times per link. In the following, we show that in the end-to-end data propagation delay analysis of the ST messages, we can omit the activation times of the links in the path of the message except for the last link. This reduction in the timed-path significantly decreases the number of timed-paths to evaluate and in turn reduces the computation time of the analysis. We show this with the following lemma.

Lemma 1. *It is enough to consider the activation time of an ST message on its last link between its sender and receiver end-stations when extracting the reachable timed-paths through the TSN network.*

Proof. Response-time analysis for ST traffic is not required as the ST traffic is scheduled offline. Feasible TSN schedules for ST class guarantee by construction the schedulability of the ST traffic. In addition, a feasible TSN schedule ensures that during the transmission of the ST message until its arrival to the destination end-station, there will be no new activation of the subsequent instances of the ST message. The activations and arrivals of ST frames within the network are known

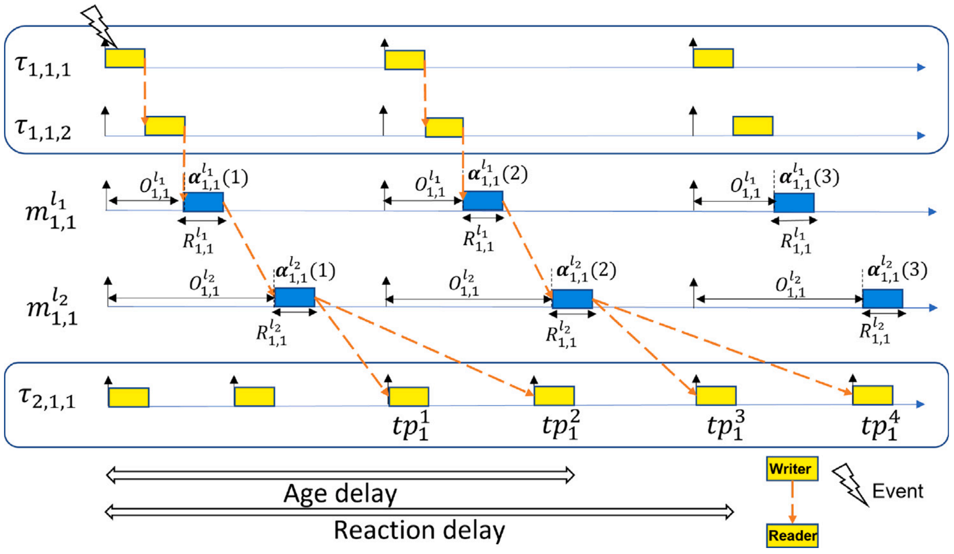


Fig. 12. Timed-path with ST messages.

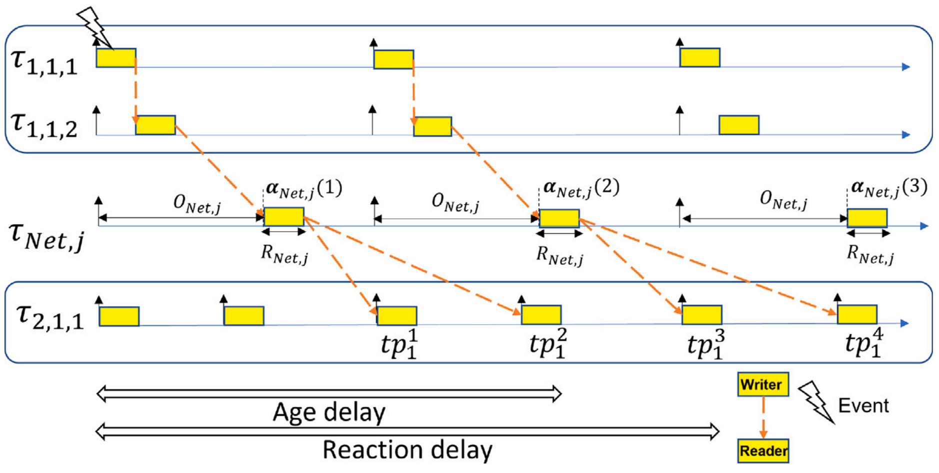


Fig. 13. Modeling the ST message at the last hop.

prior to the end-to-end analysis, thus there is always a deterministic reachable timed-path for an ST message over several links, which is already calculated and is configured offline, at the time of scheduling the ST traffic. Because the reachability of this deterministic path is already approved with the offline scheduling algorithm, it is only required to check the reachability of this path to the receiver node. Therefore, it is enough to consider the activation of the ST message on the last link (the end of the network's reachable path) when extracting the reachable timed-paths.

Based on Lemma 1 in case of the scenario depicted in Fig. 12, the message at its last hop ($m^{l_2}_{1,1}$) is enough to consider in the identification of reachable timed-paths for the end-to-end data propagation delay analysis, which is shown in Fig. 13. Hence, the existing analysis needs to be extended in order to support the TSN classes.

According to Lemma 1, only considering the ST message activation on its last link is sufficient for deriving the timed-path. However to consider the ST message on the last link, we propose to model the whole path of the ST message as a separate task which simplifies the incorporation of the ST message in the existing analysis. Eq. (21) shows the model of such a task. We regard this task as the network task and denote it by $\tau_{Net,j}$. This task corresponds to the message m^r_{jk}. The parameter r is the ID of the link delivering the message to the receiving end-station (last hop). Therefore, each ST message is modeled with a $\tau_{Net,j}$ task.

$$\tau_{Net,j} := \left\langle P_{Net,j} = P_{jk}, C_{Net,j} = TT(Size_{jk}), T_{Net,j} = T_{jk}, J_{Net,j} = J_{jk}, O_{Net,j} = O^r_{jk} \right\rangle \tag{21}$$

where, $TT()$ calculates the transmission time of the message based on the size of the message and the network speed (s). $TT()$ is calculated by $((Size_{jk} + OH) * 8)/(s)$. Where, OH is the TSN frame's overhead in Bytes.

Modeling of the ST message with a task allows us to use Eq. (20) to evaluate the reachability of timed-paths corresponding to a transaction, where $\tau_{Net,j}$ can be a reader or writer task in the transaction j, while its activation time is computed according to Eq. (10).

5.3. Accounting for the non-ST messages

A non-ST message is assumed to be scheduled for transmission as soon as the sender task completes its execution. The arrival time of the message instance to the reader task instance is calculated based on the activation time and the period of the writer task instance and the response time of the message. As depicted in Fig. 14, the non-ST

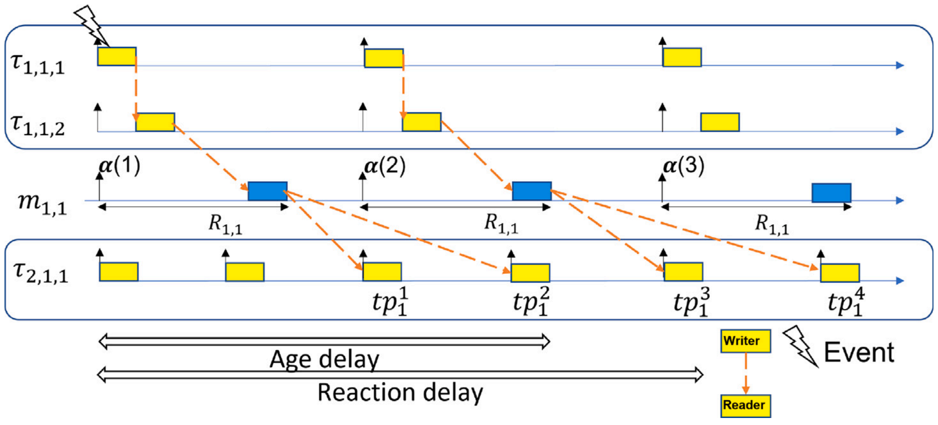


Fig. 14. Timed-path with non-ST messages.

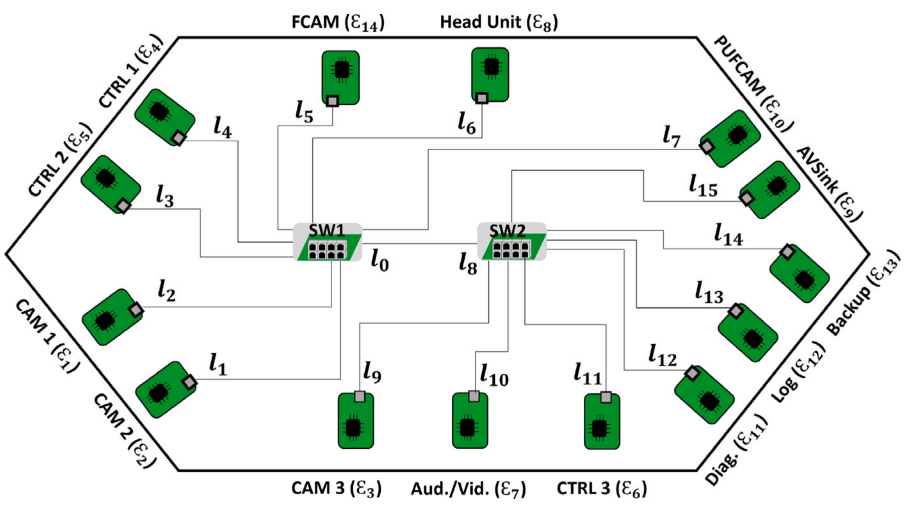


Fig. 15. Vehicular application use case.

message $m_{1,1}$ has no offset in the set of two links in its route to the reader task, namely $\tau_{2,1,1}$. As a result, the activation time of the message is assumed to be the same as its predecessor writer task. In Fig. 14, each instance of $m_{1,1}$ is not transmitted immediately after the completion of the sender task $\tau_{1,1,2}$ because we assume the message received interference from other higher priority messages (and/or it was blocked by the lower priority messages). By knowing the worst-case response time of the message ($R_{1,1}$) the reachable instance of the reader task can be determined. For example, consider tp_1^1 and tp_1^2 in Fig. 14. In these timed-paths, the first instance of the message is reachable to the third and fourth instance of the reader task. Therefore, Eq. (17) is used to evaluate the reachability of timed-paths in a transaction.

5.4. Calculation of worst-case data age and reaction time delays

The data age and reaction time delays are derived based on timed-paths (introduced in Section 5.1). In this section, we further explain the calculation of the worst-case data age and reaction time delays according to [5]. The data age delay for a timed-path tp_j^n belonging to the transaction Γ_j is calculated by Eq. (22), which calculates the time difference between the input data and the last sample of the corresponding output data.

$$\Delta_{age}(tp_j^n) = \alpha_{last}(tp_j^n) + RT_{last}(tp_j^n) - \alpha_{first}(tp_j^n) \tag{22}$$

where, $\alpha_{first}()$ returns the activation time of the task instance that is the first task receiving the fresh input data. This task is an instance of the transaction's initiator task inside the initiator end-station. Also, $\alpha_{last}()$ and $RT_{last}()$ return the activation time and the worst-case response time of the instance of the terminator task from the terminator end-station, after which the data is overwritten.

The reaction time delay is calculated by Eq. (23), where $Pred()$ represents the first instance of the timed-path before the timed-path under analysis tp_j^n. Note that the effect of just missing an event at the input of the task chain is covered by $Pred()$.

$$\Delta_{reac}(tp_j^n) = \alpha_{last}(tp_j^n) + RT_{last}(tp_j^n) - \alpha_{first}(Pred(tp_j^n)) \tag{23}$$

The data age and reaction time delays for all timed-paths for every transaction should be extracted and the longest corresponding values represent the worst-case data age and reaction delays, which are calculated according to Eq. (24).

$$\begin{aligned} \Delta_{age}(\Gamma_j) &= \{max(\Delta_{age}(tp_j^n)); n = 1...z\} \\ \Delta_{reac}(\Gamma_j) &= \{max(\Delta_{reac}(tp_j^n)); n = 1...z\} \end{aligned} \tag{24}$$

The instances of the task set in a periodic system repeat in a given bounded pattern specified by the LCM of the periods of the tasks in the task set. Moreover, to find the worst-case data age and reaction time delays, it is sufficient to enumerate and compare all timed-paths within a finite bound of twice the hyperperiod (LCM of all the involved periods in the transaction) [5].

It is conventionally desired that the data age and reaction time delays are less than or equal to their corresponding constraints according to Eq. (25).

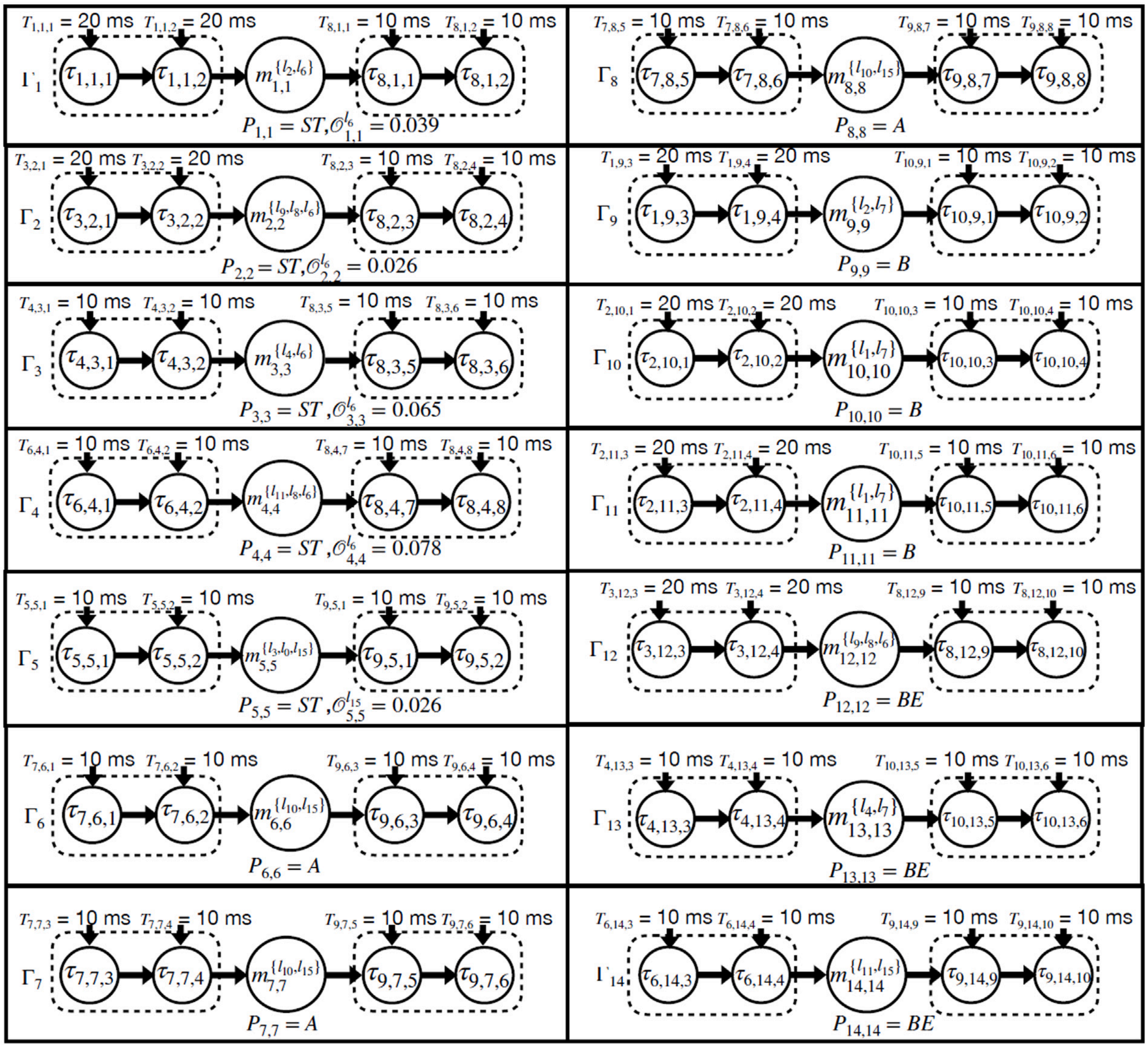

$T_{1,1,1}$ = 20 ms $T_{1,1,2}$ = 20 ms $T_{8,1,1}$ = 10 ms $T_{8,1,2}$ = 10 ms
Γ_1 $\tau_{1,1,1}$ $\tau_{1,1,2}$ $m_{1,1}^{\{l_2,l_6\}}$ $\tau_{8,1,1}$ $\tau_{8,1,2}$
$P_{1,1} = ST, \mathcal{O}_{1,1}^{l_6} = 0.039$
$T_{3,2,1}$ = 20 ms $T_{3,2,2}$ = 20 ms $T_{8,2,3}$ = 10 ms $T_{8,2,4}$ = 10 ms
Γ_2 $\tau_{3,2,1}$ $\tau_{3,2,2}$ $m_{2,2}^{\{l_9,l_8,l_6\}}$ $\tau_{8,2,3}$ $\tau_{8,2,4}$
$P_{2,2} = ST, \mathcal{O}_{2,2}^{l_6} = 0.026$
$T_{4,3,1}$ = 10 ms $T_{4,3,2}$ = 10 ms $T_{8,3,5}$ = 10 ms $T_{8,3,6}$ = 10 ms
Γ_3 $\tau_{4,3,1}$ $\tau_{4,3,2}$ $m_{3,3}^{\{l_4,l_6\}}$ $\tau_{8,3,5}$ $\tau_{8,3,6}$
$P_{3,3} = ST, \mathcal{O}_{3,3}^{l_6} = 0.065$
$T_{6,4,1}$ = 10 ms $T_{6,4,2}$ = 10 ms $T_{8,4,7}$ = 10 ms $T_{8,4,8}$ = 10 ms
Γ_4 $\tau_{6,4,1}$ $\tau_{6,4,2}$ $m_{4,4}^{\{l_{11},l_8,l_6\}}$ $\tau_{8,4,7}$ $\tau_{8,4,8}$
$P_{4,4} = ST, \mathcal{O}_{4,4}^{l_6} = 0.078$
$T_{5,5,1}$ = 10 ms $T_{5,5,2}$ = 10 ms $T_{9,5,1}$ = 10 ms $T_{9,5,2}$ = 10 ms
Γ_5 $\tau_{5,5,1}$ $\tau_{5,5,2}$ $m_{5,5}^{\{l_3,l_0,l_{15}\}}$ $\tau_{9,5,1}$ $\tau_{9,5,2}$
$P_{5,5} = ST, \mathcal{O}_{5,5}^{l_{15}} = 0.026$
$T_{7,6,1}$ = 10 ms $T_{7,6,2}$ = 10 ms $T_{9,6,3}$ = 10 ms $T_{9,6,4}$ = 10 ms
Γ_6 $\tau_{7,6,1}$ $\tau_{7,6,2}$ $m_{6,6}^{\{l_{10},l_{15}\}}$ $\tau_{9,6,3}$ $\tau_{9,6,4}$
$P_{6,6} = A$
$T_{7,7,3}$ = 10 ms $T_{7,7,4}$ = 10 ms $T_{9,7,5}$ = 10 ms $T_{9,7,6}$ = 10 ms
Γ_7 $\tau_{7,7,3}$ $\tau_{7,7,4}$ $m_{7,7}^{\{l_{10},l_{15}\}}$ $\tau_{9,7,5}$ $\tau_{9,7,6}$
$P_{7,7} = A$
$T_{7,8,5}$ = 10 ms $T_{7,8,6}$ = 10 ms $T_{9,8,7}$ = 10 ms $T_{9,8,8}$ = 10 ms
Γ_8 $\tau_{7,8,5}$ $\tau_{7,8,6}$ $m_{8,8}^{\{l_{10},l_{15}\}}$ $\tau_{9,8,7}$ $\tau_{9,8,8}$
$P_{8,8} = A$
$T_{1,9,3}$ = 20 ms $T_{1,9,4}$ = 20 ms $T_{10,9,1}$ = 10 ms $T_{10,9,2}$ = 10 ms
Γ_9 $\tau_{1,9,3}$ $\tau_{1,9,4}$ $m_{9,9}^{\{l_2,l_7\}}$ $\tau_{10,9,1}$ $\tau_{10,9,2}$
$P_{9,9} = B$
$T_{2,10,1}$ = 20 ms $T_{2,10,2}$ = 20 ms $T_{10,10,3}$ = 10 ms $T_{10,10,4}$ = 10 ms
Γ_{10} $\tau_{2,10,1}$ $\tau_{2,10,2}$ $m_{10,10}^{\{l_1,l_7\}}$ $\tau_{10,10,3}$ $\tau_{10,10,4}$
$P_{10,10} = B$
$T_{2,11,3}$ = 20 ms $T_{2,11,4}$ = 20 ms $T_{10,11,5}$ = 10 ms $T_{10,11,6}$ = 10 ms
Γ_{11} $\tau_{2,11,3}$ $\tau_{2,11,4}$ $m_{11,11}^{\{l_1,l_7\}}$ $\tau_{10,11,5}$ $\tau_{10,11,6}$
$P_{11,11} = B$
$T_{3,12,3}$ = 20 ms $T_{3,12,4}$ = 20 ms $T_{8,12,9}$ = 10 ms $T_{8,12,10}$ = 10 ms
Γ_{12} $\tau_{3,12,3}$ $\tau_{3,12,4}$ $m_{12,12}^{\{l_9,l_8,l_6\}}$ $\tau_{8,12,9}$ $\tau_{8,12,10}$
$P_{12,12} = BE$
$T_{4,13,3}$ = 10 ms $T_{4,13,4}$ = 10 ms $T_{10,13,5}$ = 10 ms $T_{10,13,6}$ = 10 ms
Γ_{13} $\tau_{4,13,3}$ $\tau_{4,13,4}$ $m_{13,13}^{\{l_4,l_7\}}$ $\tau_{10,13,5}$ $\tau_{10,13,6}$
$P_{13,13} = BE$
$T_{6,14,3}$ = 10 ms $T_{6,14,4}$ = 10 ms $T_{9,14,9}$ = 10 ms $T_{9,14,10}$ = 10 ms
Γ_{14} $\tau_{6,14,3}$ $\tau_{6,14,4}$ $m_{14,14}^{\{l_{11},l_{15}\}}$ $\tau_{9,14,9}$ $\tau_{9,14,10}$
$P_{14,14} = BE$

Fig. 16. Evaluation settings (chains of tasks).

$$\begin{aligned} &\Delta_{age}(\Gamma_j) <= Age_j \\ &\Delta_{reac}(\Gamma_j) <= Reac_j \end{aligned} \tag{25}$$

After the analysis the values of the data age and reaction time delays can be examined to check if they satisfy the specific needs of the targeted system according to the user-defined end-to-end timing constraints, i.e., based on different criteria as explained in the works [16,48].

6. Vehicular application case study

In this section, we discuss a vehicular industrial use case that is used to evaluate the presented end-to-end data propagation delay analysis. The use case consists of 14 end-stations that are connected by a two-switch TSN network as illustrated in Fig. 15. Each end-station is assumed to include multiple tasks. Fig. 15 is inspired by a use case developed in [53] and the traffic is specified accordingly. We implemented the proposed analysis as an in-house tool. The configuration and message set in the use case were given as input to the implemented analysis (see Fig. 15).

6.1. Experimental setup

We perform the extended as well as the existing end-to-end data propagation delay analysis of the use case in Fig. 15. The evaluation of the analyses is performed under different traffic scenarios given to the use case. In this experiment, we assume there are 14 transactions starting from end-stations 1 to 7 that use different TSN traffic classes to communicate with three sink end-stations with the IDs 8 to 10. Table 1 shows the transaction settings. Moreover, Fig. 16 illustrates the relation between the elements of the transactions shown in Table 1. Note that in Fig. 16, we do not show registers between any two messages within a node or two end-stations for the sake of enhancing readability of the figure. Each transaction initiates and terminates by a task within different end-stations. Each transaction includes four tasks in total. Besides, each transaction includes two tasks per end-station which constitute the chain. In the source end-station, the first task is a computation task and the subsequent task is a communication task, which receives data input from the predecessor task, then prepares and injects messages to the network. In the destination end-station of these transactions, the first task is a communication task that receives and processes the message from the network. The communication task sends the message for further processing to the last task in the transaction. The periods of the tasks are chosen based on the automotive data set

Table 1
Evaluation settings for the use case based on distributed chains.

Γ_j	$\mathcal{E}_i$	Source tasks (τ_{ijk}): [id, P_{ijk}, C_{ijk}, T_{ijk}]		Message (m_{jk}): [id, P_{jk}, $Size_{jk}$, T_{jk}, $\mathcal{O}^r_{jk}$]	$\mathcal{E}_i$	Destination tasks (τ_{ijk}): [id, P_{ijk}, C_{ijk}, T_{ijk}]	
		Computation	Communication			Communication	Computation
1	$\mathcal{E}_1$	[$\tau_{1,1,1}$,4,0.5,20]	[$\tau_{1,1,2}$,3,0.5,20]	[$m_{1,1}$, ST,1500,20,1.039]	$\mathcal{E}_8$	[$\tau_{8,1,1}$,10,0.5,10]	[$\tau_{8,1,2}$,9,0.5,10]
2	$\mathcal{E}_3$	[$\tau_{3,2,1}$,4,0.5,20]	[$\tau_{3,2,2}$,3,0.5,20]	[$m_{2,2}$, ST,1500,20,1.026]	$\mathcal{E}_8$	[$\tau_{8,2,3}$,8,0.5,10]	[$\tau_{8,2,4}$,7,0.5,10]
3	$\mathcal{E}_4$	[$\tau_{4,3,1}$,4,0.5,10]	[$\tau_{4,3,2}$,3,0.5,10]	[$m_{3,3}$, ST,1500,10,1.065]	$\mathcal{E}_8$	[$\tau_{8,3,5}$,6,0.5,10]	[$\tau_{8,3,6}$,5,0.5,10]
4	$\mathcal{E}_6$	[$\tau_{6,4,1}$,4,0.5,10]	[$\tau_{6,4,2}$,3,0.5,10]	[$m_{4,4}$, ST,1500,10,1.078]	$\mathcal{E}_8$	[$\tau_{8,4,7}$,4,0.5,10]	[$\tau_{8,4,8}$,3,0.5,10]
5	$\mathcal{E}_5$	[$\tau_{5,5,1}$,2,0.5,10]	[$\tau_{5,5,2}$,1,0.5,10]	[$m_{5,5}$, ST,1500,10,1.052]	$\mathcal{E}_9$	[$\tau_{9,5,1}$,10,0.5,10]	[$\tau_{9,5,2}$,9,0.5,10]
6	$\mathcal{E}_7$	[$\tau_{7,6,1}$,6,0.5,10]	[$\tau_{7,6,2}$,5,0.5,10]	[$m_{6,6}$, A,1500,10,0]	$\mathcal{E}_9$	[$\tau_{9,6,3}$,8,0.5,10]	[$\tau_{9,6,4}$,7,0.5,10]
7	$\mathcal{E}_7$	[$\tau_{7,7,3}$,4,0.5,10]	[$\tau_{7,7,4}$,3,0.5,10]	[$m_{7,7}$, A,1500,10,0]	$\mathcal{E}_9$	[$\tau_{9,7,5}$,6,0.5,10]	[$\tau_{9,7,6}$,5,0.5,10]
8	$\mathcal{E}_7$	[$\tau_{7,8,5}$,2,0.5,10]	[$\tau_{7,8,6}$,1,0.5,10]	[$m_{8,8}$, A,1500,10,0]	$\mathcal{E}_9$	[$\tau_{9,8,7}$,4,0.5,10]	[$\tau_{9,8,8}$,3,0.5,10]
9	$\mathcal{E}_1$	[$\tau_{1,9,3}$,2,0.5,20]	[$\tau_{1,9,4}$,1,0.5,20]	[$m_{9,9}$, B,1500,20,0]	$\mathcal{E}_{10}$	[$\tau_{10,9,1}$,8,0.5,10]	[$\tau_{10,9,2}$,7,0.5,10]
10	$\mathcal{E}_2$	[$\tau_{2,10,1}$,4,0.5,20]	[$\tau_{2,10,2}$,3,0.5,20]	[$m_{10,10}$, B,1500,20,0]	$\mathcal{E}_{10}$	[$\tau_{10,10,3}$,6,0.5,10]	[$\tau_{10,10,4}$,5,0.5,10]
11	$\mathcal{E}_2$	[$\tau_{2,11,3}$,2,0.5,20]	[$\tau_{2,11,4}$,1,0.5,20]	[$m_{11,11}$, B,1500,20,0]	$\mathcal{E}_{10}$	[$\tau_{10,11,5}$,4,0.5,10]	[$\tau_{10,11,6}$,3,0.5,10]
12	$\mathcal{E}_3$	[$\tau_{3,12,3}$,2,0.5,20]	[$\tau_{3,12,4}$,1,0.5,20]	[$m_{12,12}$, BE,1500,20,0]	$\mathcal{E}_8$	[$\tau_{8,12,9}$,2,0.5,10]	[$\tau_{8,12,10}$,1,0.5,10]
13	$\mathcal{E}_4$	[$\tau_{4,13,3}$,2,0.5,10]	[$\tau_{4,13,4}$,1,0.5,10]	[$m_{13,13}$, BE,1500,10,0]	$\mathcal{E}_{10}$	[$\tau_{10,13,7}$,2,0.5,10]	[$\tau_{10,13,8}$,1,0.5,10]
14	$\mathcal{E}_6$	[$\tau_{6,14,3}$,2,0.5,10]	[$\tau_{6,14,4}$,1,0.5,10]	[$m_{14,14}$, BE,1500,10,0]	$\mathcal{E}_9$	[$\tau_{9,14,9}$,2,0.5,10]	[$\tau_{9,14,10}$,1,0.5,10]

Table 2
idleSlope of class A and class B per link.

idleSlope (Mbps)	l_1	l_2	l_7	l_{10}	l_{15}
Class A	–	–	–	0.78	0.78
Class B	0.4	0.4	0.4	–	–

Table 3
Timing constraints for all transactions.

	Reac$_j$ (ms)	Age$_j$ (ms)
Cr$_j$	35	25

Table 4
Response times of the tasks and messages in the transactions (ms).

Trans. (Γ_j)	R_{ijk} of sender	R_{ij} of message	R_{ijk} of last task
1	$R_{1,1,2} = 1$	$R_{1,1} = 0.025$	$R_{8,1,2} = 1$
2	$R_{3,2,2} = 1$	$R_{2,2} = 0.038$	$R_{8,2,4} = 2$
3	$R_{4,3,2} = 1$	$R_{3,3} = 0.025$	$R_{8,3,6} = 3$
4	$R_{6,4,2} = 1$	$R_{4,4} = 0.038$	$R_{8,4,8} = 4$
5	$R_{5,5,2} = 1$	$R_{5,5} = 0.038$	$R_{9,5,2} = 1$
6	$R_{7,6,2} = 1$	$R_{6,6} = 1.081$	$R_{9,6,4} = 2$
7	$R_{7,7,4} = 2$	$R_{7,7} = 2.081$	$R_{9,7,6} = 3$
8	$R_{7,8,6} = 3$	$R_{8,8} = 3.081$	$R_{9,8,8} = 4$
9	$R_{1,9,4} = 2$	$R_{9,9} = 2.218$	$R_{10,9,2} = 1$
10	$R_{2,10,2} = 1$	$R_{10,10} = 1.262$	$R_{10,10,4} = 2$
11	$R_{2,11,4} = 2$	$R_{11,11} = 2.262$	$R_{10,11,6} = 3$
12	$R_{3,12,4} = 2$	$R_{12,12} = 2.127$	$R_{8,12,10} = 5$
13	$R_{4,13,4} = 2$	$R_{13,13} = 3.150$	$R_{10,13,8} = 4$
14	$R_{6,14,4} = 2$	$R_{14,14} = 2.398$	$R_{9,14,10} = 5$

in [54], i.e., $\{1, 2, 5, 10, 20, 50, 100\}$ in milliseconds (ms). The size of all messages are fixed to 1500 $Bytes$ as the maximum payload size and the worst-case execution time of each task (WCET) is considered to be 0.5 ms. The end-stations run their tasks according to the fixed-priority preemptive scheduling algorithm. In each transaction, we assume that the priority of a task is higher than the priority of its subsequent task within the same end-station. Because, we want to keep the execution of two subsequent tasks inside the same end-station in such an order to avoid creating delay in writing of data. This ensures that each precedent task (writer task) in the transaction must be executed before the subsequent task in the chain (reader task) in the case if the two tasks are activated for execution at the same time.

Five out of fourteen transactions use ST traffic class; three transactions use class A; three transactions use class B; and three transactions use class BE. The CBS mechanism is set according to Table 2, where the credit for classes A and B are chosen according to the utilization of these classes on the network links. The overall network speed is set to 1 $Gbps$. We set the idle slope (*idleSlope*) according to the utilization of the traffic on classes A and B, as shown in Table 2. The transactions 6, 7 and 8 use class A on the links 10 and 15, therefore the credit of the links l_{10} and l_{15} are set to 0.78. Furthermore, the transactions 9, 10 and 11 use the class B on the links 1, 2 and 7. Accordingly, the credits for class B on the links l_1, l_2 and l_7 are set to 0.4. We note that zero credit means there are no messages from the associated CBS classes on the link. In Table 2 only the credit for the links utilizing CBS is shown. Finally, the data age and reaction time constraints specified on each transaction are depicted in Table 3.

6.2. Evaluation of the existing and extended analyses

In this section, we compare the analysis results acquired by performing the existing [5] and the extended end-to-end data propagation delay analyses. Table 4 shows the response times of each message ($R_{i,j}$) and its sending task ($R_{i,j,k}$) separately. Moreover, the response time of the last task in each transaction is shown in Table 4.

The data age and reaction time delays for each individual transaction depicted in Table 1 are calculated with the existing and extended end-to-end data-propagation delay analyses. The analyses results are shown in Table 5.

It can be observed in Table 5 that the reaction time and data age delays calculated by the extended end-to-end data-propagation delay analysis are smaller than those calculated by the existing analysis. For example, the reaction time and data age delays of transaction Γ_5 calculated with the existing analysis are 22.09 ms and 12.09 ms respectively. Whereas, the reaction time and data age delays of transaction Γ_5 calculated with the extended analysis are 21 ms and 11 ms respectively. We note that the existing analysis calculates the reaction time and data age delays of Γ_5 with 5.19% and 9.90% over-estimation (pessimism) compared to the extended analysis. We visually demonstrate the reaction time and data age delays in Γ_5 with the help of execution traces in Fig. 17. In Γ_5, the tasks in the sender end-station ($\mathcal{E}_5$) and the receiver end-station ($\mathcal{E}_9$) are synchronized for using the ST class. The extended analysis considers synchronization of the nodes in Γ_5, i.e., each node sees the same time 0. Fig. 17(a) shows the reachable timed-paths within twice the hyperperiod (LCM of the periods of all tasks in Γ_5, which

Table 5
Calculated reaction time and data age delays for each transaction.

	Results with the existing analysis		Results with the extended analysis	
Trans. (Γ_j)	Reaction delay (ms)	Age delay (ms)	Reaction delay (ms)	Age delay (ms)
1	32.064	22.064	31	21
2	33.064	23.064	32	22
3	24.09	14.09	23	13
4	25.116	15.116	24	14
5	22.09	12.09	21	11
6	23.081	13.081	22	12
7	25.081	15.081	23	13
8	27.081	17.081	24	14
9	33.218	23.218	31	21
10	33.262	23.262	32	22
11	35.262	25.262	33	23
12	37.127	27.127	35	25
13	27.15	17.15	24	14
14	27.398	17.398	25	15

Table 6
The transaction specifications.

Γ_d	$\mathcal{E}_i$	Source tasks (τ_{ijk}): [id, P_{ijk}, C_{ijk}, T_{ijk}]	Message (m_{jk}): [id, P_{jk}, $Size_{jk}$, T_{jk}, $\mathcal{O}^r_{jk}$]	$\mathcal{E}_i$	Destination tasks (τ_{ijk}): [id, P_{ijk}, C_{ijk}, T_{ijk}]	
		Task 1			Task 2	Task 3
1	$\mathcal{E}_3$	[$\tau_{1,1,1}$,1,1,50]	[$m_{1,1}$, ST,1500,50,X]	$\mathcal{E}_8$	[$\tau_{9,1,1}$,2,1,Y]	[$\tau_{9,1,2}$,1,1,Y]

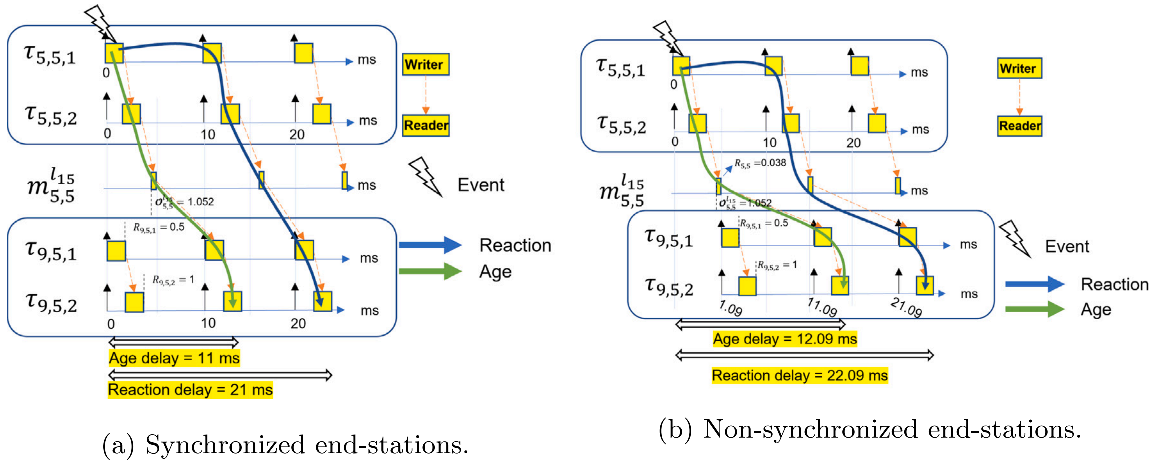


Fig. 17. Demonstration of data age and reaction time delays in Γ_5 in Table 5 with execution traces.

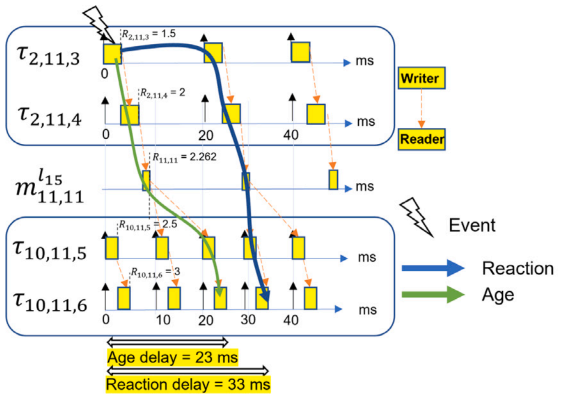
is 20 ms). On the other hand, the execution trace considered by the existing analysis is shown in Fig. 17(b).

Similarly, consider another transaction, Γ_{11}, in Table 5. The reaction time and data age delays of transaction Γ_{11} calculated with the existing analysis are subsequently 35.262 ms and 25.262 ms. Whereas, these delays calculated with the extended analysis are 33 ms and 23 ms respectively. Hence, the existing analysis calculates the reaction time and data age delays of Γ_{11} with 6.85% and 9.83% over-estimation (pessimism) compared to the extended analysis. The reaction time and data age delays in Γ_{11} are visually demonstrated with the help of execution traces in Fig. 18. Γ_{11} uses a class B message. The extended analysis considers synchronization of the nodes in Γ_{11}, i.e., each node sees the same time 0 as shown in the execution trace in Fig. 18(a). Whereas, the execution trace considered by the existing analysis is shown in Fig. 18(b).

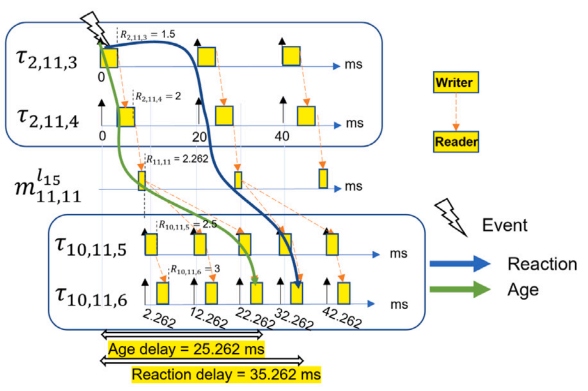
6.3. Impact of various parameters on the data age and reaction time delays

In this subsection, we demonstrate the impact of various parameters on the data age and reaction time delays using the extended end-to-end data propagation delay analysis. The parameters of interest include the offset of the ST messages, and the periods of the tasks in the receiver end-stations (i.e., the periods of the message receiving tasks). The evaluations are carried out in a transaction in the use case shown in Fig. 15. The transaction's details are shown in Table 6.

The transaction under analysis starts from CAM1 as the sending end-station and finishes in AVSink as the receiving end-station. This transaction uses one task in CAM1 with the fixed period of 50 ms and the WCET of 1 ms. CAM1 sends an ST message that is routed to AVSink via three links, namely l_2, l_0 and l_{15} according to the topology shown in Fig. 15. There are two tasks in the AVSink that are engaged in this transaction, namely $\tau_{9,1,1}$ and $\tau_{9,1,2}$. Each of these tasks has the WCET of



(a) Synchronized end-stations.



(b) Non-synchronized end-stations.

Fig. 18. Demonstration of data age and reaction time delays in Γ_{11} in Table 5 with execution traces.

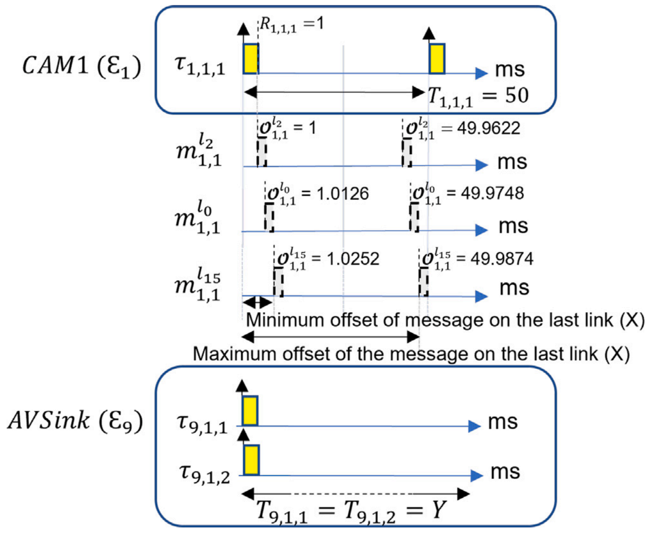

$CAM1\ (\mathcal{E}_1)$
$\tau_{1,1,1}$
$R_{1,1,1}$ = 1
$T_{1,1,1} = 50$
$m^{l_2}_{1,1}$
$O^{l_2}_{1,1}$ = 1
$O^{l_2}_{1,1}$ = 49.9622
$m^{l_0}_{1,1}$
$O^{l_0}_{1,1}$ = 1.0126
$O^{l_0}_{1,1}$ = 49.9748
$m^{l_{15}}_{1,1}$
$O^{l_{15}}_{1,1}$ = 1.0252
$O^{l_{15}}_{1,1}$ = 49.9874
ms
Minimum offset of message on the last link (X)
Maximum offset of the message on the last link (X)
$AVSink\ (\mathcal{E}_9)$
$\tau_{9,1,1}$
$\tau_{9,1,2}$
$T_{9,1,1} = T_{9,1,2} = Y$

Fig. 19. Demonstration of the minimum and maximum values of the offsets on the last link for the message under analysis.

1 ms. The task $\tau_{9,1,1}$ in AVSink that receives messages from the TSN network has the highest priority. We vary the period of the receiver end-station's tasks (AVSink) shown by notation Y in Table 6. Y can obtain values from the range $\{15, 20, 25, 30, 40, 50, 60, 70, 90, 100, 150, 200, 250\}$ in milliseconds (ms). Both tasks of AVSink have the same period.

As we consider an ST message in the transaction under analysis, we can assign any value to its offset between the range of *minimum offset* and *maximum offset* on its last link $m^{l_{15}}_{1,1}$. The variable value of the offset is denoted by X notation in Table 6. The minimum offset of the message on the last link ($m^{l_{15}}_{1,1}$) is equal to the sum of the response time of the sender task, $\tau_{1,1,1}$ (1 ms), transmission time of the message on the first link $m^{l_2}_{1,1}$ (0.0126 ms) and the transmission time of the message on the second link $m^{l_0}_{1,1}$ (0.0126 ms) as shown in Fig. 19. The sum of these three terms is equal to 1.0252 ms, which defines the minimum value of the offset that can be assigned to the message under analysis on its last link towards the destination end-station. Similarly, the maximum value of the offset that can be assigned to the message under analysis on its last link towards the destination end-station is 49.9874 ms, which is equal to the difference between the message's period (50 ms) and transmission time of the message on the last link (0.0126 ms). There are no other TSN messages interfering with the ST message in this transaction. According to the Worst-Case response-Time Analysis of TSN [31,33,34], the response time of the message (the time the message is received at the receiver end-station) is 1.025 ms. The experiments are carried out by calculating and comparing the data age and reaction time delays, under variations made on the parameters X for ST offsets and Y for the period of the receiver end-station's tasks from the settings given in Table 6.

The data age and reaction time delays of the transaction under analysis are presented in Fig. 20 and Fig. 21 respectively. The horizontal axes in Figs. 20 and 21 show the variations in the receiver task's periods. The vertical axis represents the data age delays (in Fig. 20) and reaction time delays (in Fig. 21). Color-coded line graphs present the delays calculated based on different offsets assigned to the ST message.

Fig. 20 shows that the data age delay remains constant at 52 ms with variation in the period of the message receiving task in the transaction, when the offset of the message is set to the minimum value of 1.0252 ms. The data age delay increases as the value of the offset is increased to its maximum value of 49.9874 ms. Similarly, the reaction time delays calculated for the same transaction by setting different values of the ST message offset and by varying the period of the message receiving task within the traction are depicted in Fig. 21.

We note for some specific cases in Fig. 20, where the message receiving task's period is a harmonic multiple (50 ms, 100 ms, …) of the sender task's period (50 ms) within the transaction, the data age delay remain constant at 52 ms. Furthermore, any variation in the ST message's offset (selected between the minimum and maximum values of the offset) also does not effect the data age delay in such cases. The reason for getting the same data age delay in these specific cases is that the data age delay considers the last input in the timed path which is not overwritten by the previous inputs [5,7]. We demonstrate this situation by drawing the execution traces of the transaction in Figs. 22 and 23, where the period of the message receiving task ($\tau_{9,1,1}$) is set to the first two harmonic multiples (50 ms and 100 ms) of the period of the sender task ($\tau_{1,1,1}$, period = 50 ms). Figs. 22(a, c, e) and 23 (a, c, e) represent the execution traces of the transaction when periods of the sender task ($\tau_{1,1,1}$) and the receiver task ($\tau_{9,1,1}$) are set to 50 ms while different values of the ST message offset are considered. Whereas, Figs. 22(b, d, f) and 23(b, d, f) represent the execution traces of the transaction when periods of the sender task ($\tau_{1,1,1}$) is set to 50 ms while the period of the receiver task ($\tau_{9,1,1}$) is set to 100 ms while varying the ST message's offset.

For instance, the data age delay in Fig. 22(a) is 52 ms. Although the period of ($\tau_{9,1,1}$) is doubled in Fig. 22(b) with respect to its period in Fig. 22(a), the first instance of the message will be overwritten by the second instance of the message in Fig. 22(b). Hence, the data provided by the first instance of $\tau_{1,1,1}$ via the first instance of the message cannot

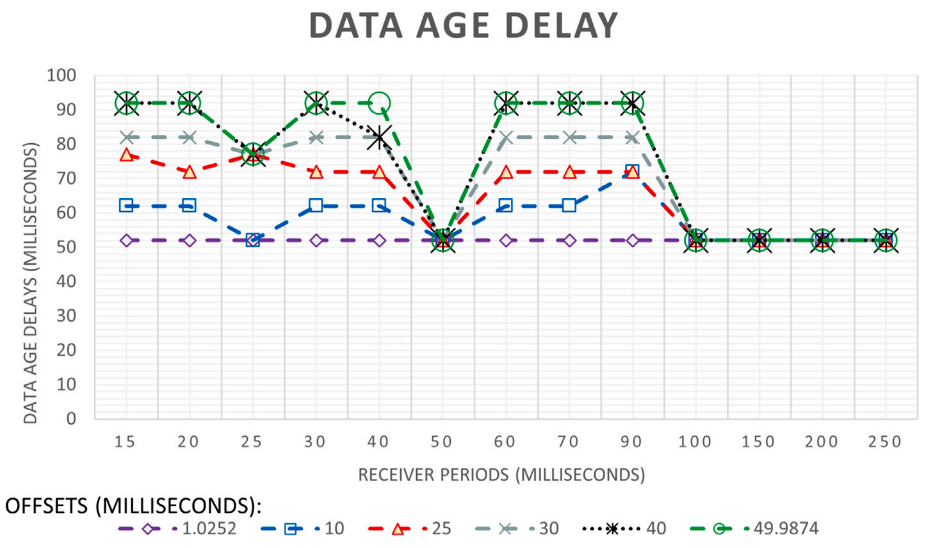


Fig. 20. Impact of variations on the message offsets and the receiver task's period on the data age delay.

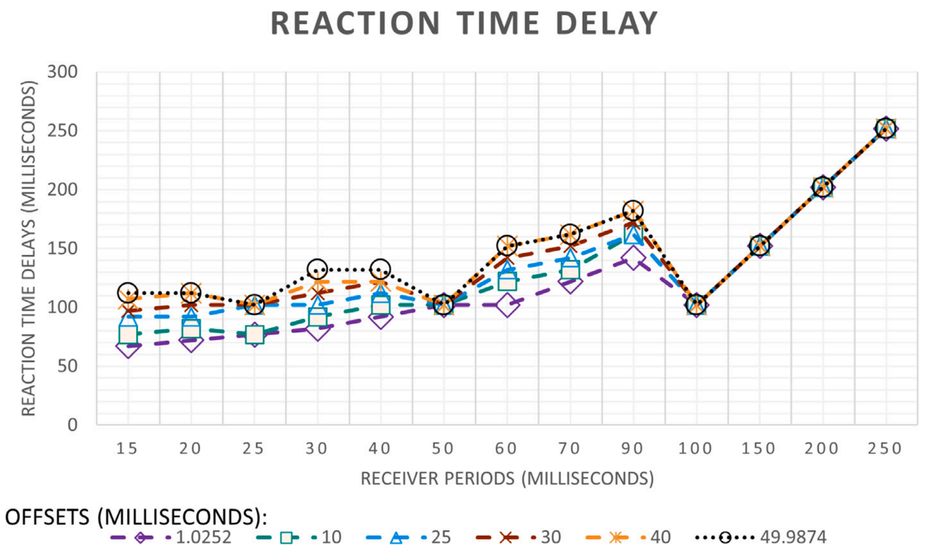


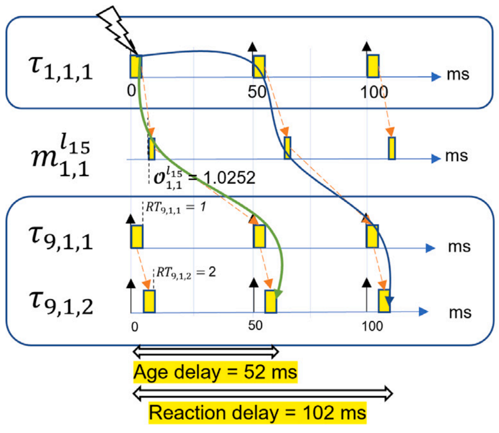
Fig. 21. Impact of variations on the message offsets and the receiver task's period on the reaction time delay.

reach the second instance of $\tau_{9,1,1}$. In fact, the second instance of $\tau_{9,1,1}$ will read the data produced by the second instance of $\tau_{1,1,1}$. Therefore, the data age delay in Fig. 22(b) is the same as that of the data age delay in Fig. 22(a). In fact, the data age delay stays the same when higher harmonic multiples of the period of $\tau_{1,1,1}$ are considered for the period of $\tau_{9,1,1}$ including 150 ms, 200 ms and 250 ms as shown in Fig. 20. Furthermore, by varying the values of the ST message's offsets, the data path for the age delay remains the same because the message offset cannot exceed the period of its sender task ($\tau_{1,1,1}$) and the period of the receiver task ($\tau_{9,1,1}$) is a harmonic multiple of the period of $\tau_{1,1,1}$.

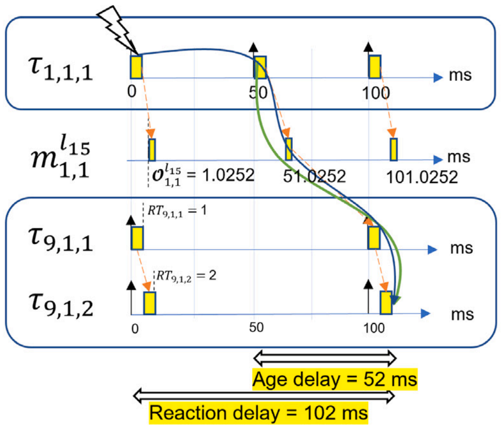
On the other hand, the reaction time delay is significantly impacted by the increase in the period of the task that is receiving the message, as shown in Fig. 21. We note that the reaction time delay is the same for the periods of $\tau_{9,1,1}$ that are the first two harmonic multiples of the period of $\tau_{1,1,1}$. However, the reaction delay keeps on increasing with the increase in the period of $\tau_{9,1,1}$ that is equal to the subsequent harmonic multiples (150 ms, 200 ms and 250 ms) of the period of $\tau_{1,1,1}$ as shown in Fig. 21.

6.4. Discussions

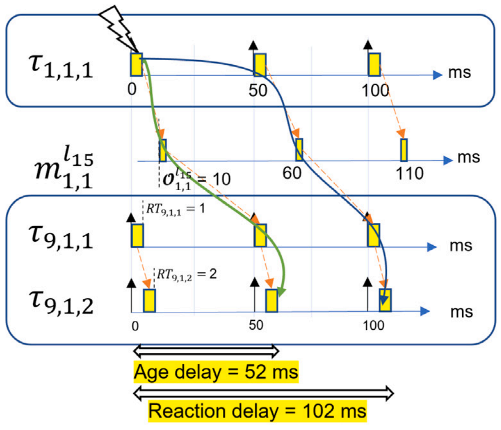
The end-to-end data propagation delays such as data age and reaction time delays calculated by the existing timing analysis can be pessimistic (over-estimated) based on different network implementations, i.e., when the sender and receiver end-stations are synchronized. Also, variations in the network configuration such as offset configuration or period of the sender and receiver tasks affect the propagation of data from the input to the output of the transaction. Hence the end-to-end data propagation delays may also vary significantly for different settings. Transactions that utilize ST class need more configuration and optimization effort due to the need for offline schedules. Though, ST classes enable flexible adjustment of the end-to-end data-propagation delays.



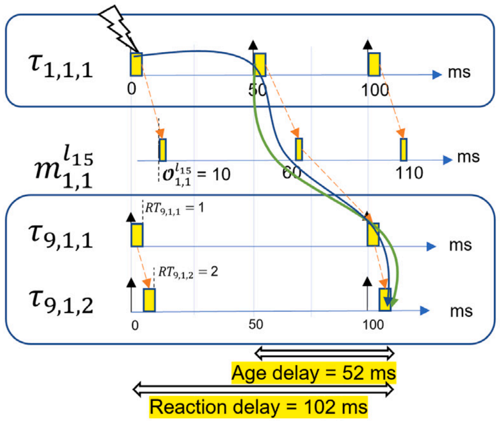
(a) $\mathcal{O}^{l_{15}}_{1,1} = 1.0252\ ms;$ $T_{9,1,1} = T_{9,1,2} = 50\ ms.$



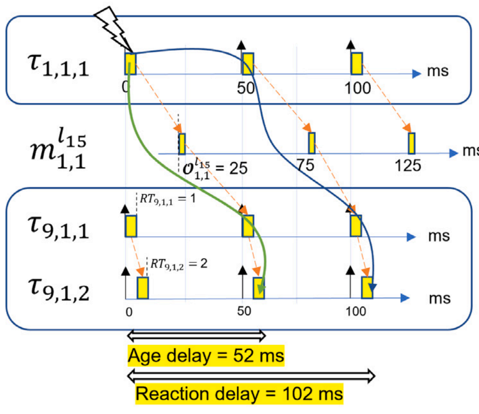
(b) $\mathcal{O}^{l_{15}}_{1,1} = 1.0252\ ms;$ $T_{9,1,1} = T_{9,1,2} = 100\ ms.$



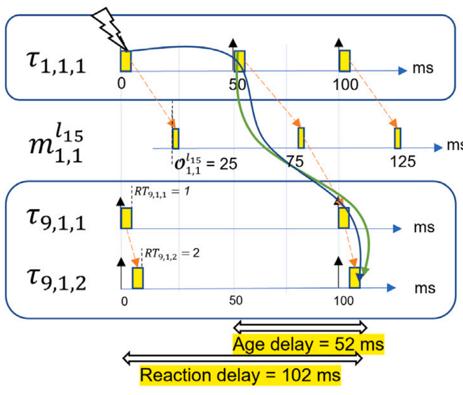
(c) $\mathcal{O}^{l_{15}}_{11} = 10\ ms;$ $T_{9,1,1} = T_{9,1,2} = 50\ ms.$



(d) $\mathcal{O}^{l_{15}}_{11} = 10\ ms;$ $T_{9,1,1} = T_{9,1,2} = 100\ ms.$

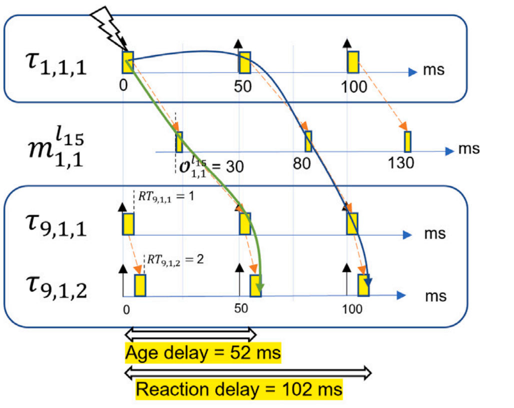


(e) $\mathcal{O}^{l_{15}}_{11} = 25\ ms;$ $T_{9,1,1} = T_{9,1,2} = 50\ ms.$

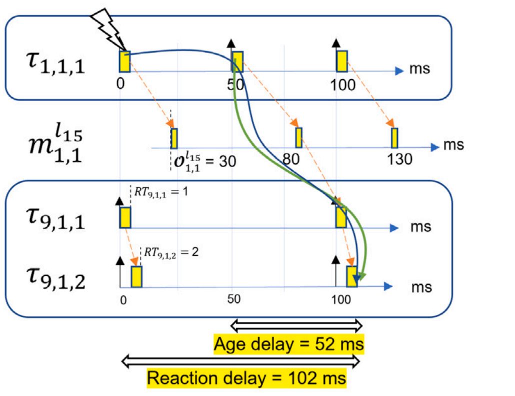


(f) $\mathcal{O}^{l_{15}}_{11} = 25\ ms;$ $T_{9,1,1} = T_{9,1,2} = 100\ ms.$

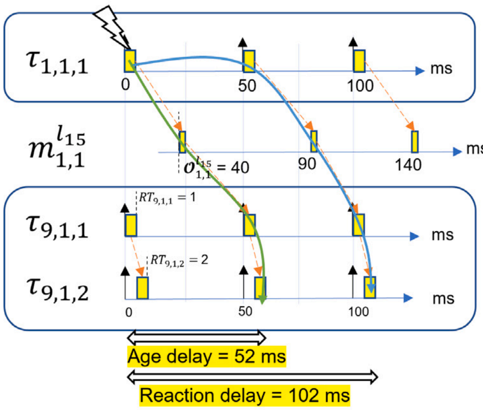
Fig. 22. The effect of offset variations and harmonically set periods on the data age and reaction time delays of the transaction (offsets 1,012 ms, 10 ms and 25 ms).



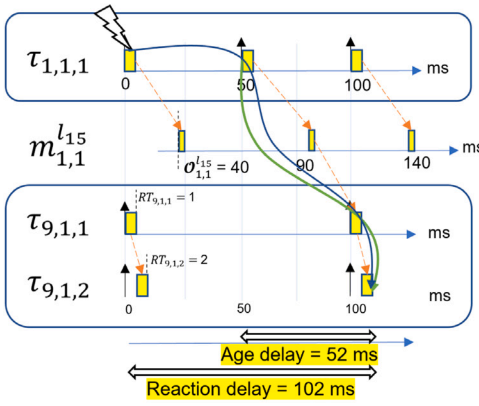
(a) $\mathcal{O}_{1,1}^{l_{15}} = 30\ ms$; $T_{9,1,1} = T_{9,1,2} = 50\ ms$.



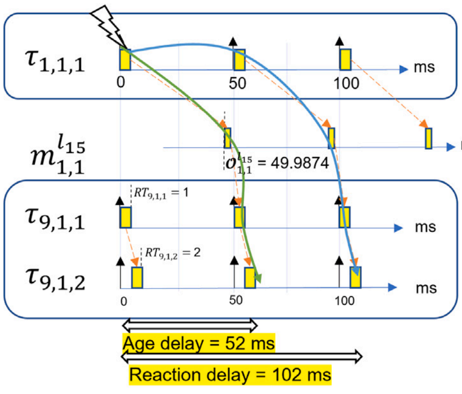
(b) $\mathcal{O}_{1,1}^{l_{15}} = 30\ ms$; $T_{9,1,1} = T_{9,1,2} = 100\ ms$.



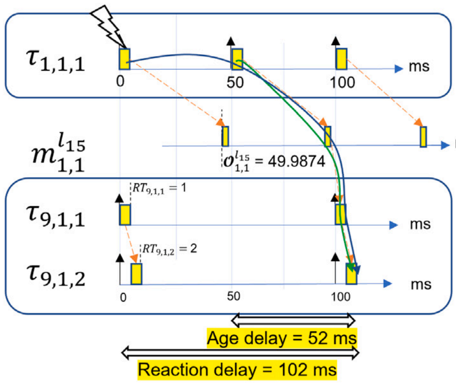
(c) $\mathcal{O}_{11}^{l_{15}} = 40\ ms$; $T_{9,1,1} = T_{9,1,2} = 50\ ms$.



(d) $\mathcal{O}_{11}^{l_{15}} = 40\ ms$; $T_{9,1,1} = T_{9,1,2} = 100\ ms$.



(e) $\mathcal{O}_{11}^{l_{15}} = 49.9874\ ms$; $T_{9,1,1} = T_{9,1,2} = 50\ ms$.



(f) $\mathcal{O}_{11}^{l_{15}} = 49.9874\ ms$; $T_{9,1,1} = T_{9,1,2} = 100\ ms$.

Fig. 23. The effect of offset variations and harmonically set periods on the data age and reaction time delays of the transaction (offsets 30 ms, 40 ms and 49.987 ms).

7. Conclusions and future works

In this paper, we presented a detailed end-to-end timing model of vehicular distributed embedded systems that utilize TSN for network communication. The end-to-end timing model enables integrating TSN features with the existing end-to-end data propagation analysis. The proposed end-to-end timing model features distributed TSN transactions over two or multiple synchronized end-stations with locally synchronized tasks. Distributed TSN transactions define a chain of tasks which can communicate within two different end-stations by TSN messages. TSN classes have different clock synchronization requirements, that have not been accounted for in the existing end-to-end data propagation delay analysis. In this paper, we proposed models and methods to incorporate all TSN traffic requirements in the end-to-end data propagation delay analysis.

The proposed end-to-end timing model and analysis were evaluated on a vehicular use case. We performed a comparative evaluation of the existing and extended analysis on the base of the use case. Moreover, the evaluations are in terms of the effects that the transaction periods and the message offsets can cause on the data age and reaction time delays.

In summary, we concluded from the results that the non-ST transactions deal with less effective parameters on the data age and reaction time delays. The data age and reaction time delays of the ST transactions are more flexibly readjusted, though the optimization with regards to their multiple configurable parameters is complex.

The results of this work additionally indicated some potential future work directions. Firstly, this work enables analysis-based optimization of the end-to-end timing delays (active approach). For example, the scheduling of the tasks within the end-stations could also be included in order to optimize the end-to-end data propagation delays. Secondly, system-level optimization of the end-to-end data propagation delays by co-scheduling of tasks and messages could be considered as extension of the work presented in this paper. Finally, a potential future work is to integrate the proposed end-to-end data propagation delay analysis with model-based software development frameworks for distributed embedded systems, e.g., Rubus-ICE [12].

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Saad Mubeen reports financial support was provided by Sweden's Innovation Agency.

Data availability

No data was used for the research described in the article.

Acknowledgments

The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) via the DESTINE, PROVIDENT and INTERCONNECT projects, and the Swedish Knowledge Foundation via the FIESTA, HERO and DPAC projects. We would like to thank our all industrial partners, especially Volvo Construction Equipment, Arcticus Systems, and HIAB.

References

[1] L. Lo Bello, R. Mariani, S. Mubeen, S. Saponara, Recent advances and trends in on-board embedded and networked automotive systems, IEEE Trans. Ind. Inform. (2019).

[2] ISO 11898-1, Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication, ISO Standard-11898, 1993.

[3] Christina Rödel, Susanne Stadler, Alexander Meschtscherjakov, Manfred Tscheligi, Towards Autonomous Cars: The Effect of Autonomy Levels on Acceptance and User Experience, ACM, 2014.

[4] 802.1Q-2022 - IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks, IEEE, 2022.

[5] N. Feiertag, K. Richter, J. Nordlander, J. Jonsson, A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics, in: Workshop on Compositional Theory and Technology for Real-Time Embedded Systems, 2008.

[6] A.C. Rajeev, Swarup Mohalik, Manoj G. Dixit, Devesh B. Chokshi, S. Ramesh, Schedulability and end-to-end latency in distributed ecu networks: formal modeling and precise estimation, in: Proceedings of the Tenth ACM International Conference on Embedded Software, EMSOFT '10, 2010.

[7] S. Mubeen, J. Mäki-Turja, M. Sjödin, Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study, Comput. Sci. Inf. Syst. (2013).

[8] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte, End-to-end timing analysis of cause-effect chains in automotive embedded systems, J. Syst. Archit. (2017).

[9] AUTOSAR Consortium, AUTOSAR Techincal Overview [online], Release 4.1, Rev.2, Ver.1.1.0., http://autosar.org.

[10] EAST-ADL Domain Model Specification, V2.1.12, http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf.

[11] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, R. Ernst, System-level performance analysis - the SymTA/S approach, Comput. Digit. Tech. (2005).

[12] S. Mubeen, H. Lawson, J. Lundbäck, M. Gålnander, K. Lundbäck, Provisioning of predictable embedded software in the vehicle industry: The Rubus approach, in: 4th International Workshop on Software Engineering Research and Industry Practice, Located at the 39th International Conference on Software Engineering, ACM, 2017.

[13] Robert Davis, Alan Burns, Reinder Bril, Johan Lukkien, Controller area network (CAN) schedulability analysis: Refuted, revisited and revised, Real-Time Syst. (2007).

[14] S. Mubeen, J. Mäki-Turja, M. Sjödin, MPS-CAN analyzer: Integrated implementation of response-time analyses for controller area network, J. Syst. Archit. (2014).

[15] Mohammad Ashjaei, Gaetano Patti, Moris Behnam, Thomas Nolte, Giuliana Alderisi, Lucia Lo Bello, Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support, Real-Time Syst. (2017).

[16] Saad Mubeen, Mohammad Ashjaei, Mikael Sjödin, Holistic modeling of time sensitive networking in component-based vehicular embedded systems, in: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, 2019.

[17] L. Lo Bello, W. Steiner, A perspective on IEEE time-sensitive networking for industrial communication and automation systems, Proc. IEEE (2019).

[18] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, S. Mubeen, Time-sensitive networking in automotive embedded systems: State-of-the-art and research opportunities, J. Syst. Archit. (2021).

[19] J. Diemer, D. Thiele, R. Ernst, Formal worst-case timing analysis of ethernet topologies with strict-priority and AVB switching, in: International Symposium on Industrial Embedded Systems, 2012.

[20] U. D. Bordoloi, A. Aminifar, P. Eles, Z. Peng, Schedulability analysis of ethernet AVB switches, in: International Conference on Embedded and Real-Time Computing Systems and Applications, 2014.

[21] X. Li, L. George, Deterministic delay analysis of AVB switched ethernet networks using an extended trajectory approach, Real-Time Syst. (2017).

[22] J. Cao, P.J.L. Cuijpers, R.J. Bril, J.J. Lukkien, Independent yet tight WCRT analysis for individual priority classes in ethernet AVB, in: International Conference on Real-Time Networks and Systems, 2016.

[23] A. Finzi, A. Mifdaoui, F. Frances, E. Lochin, Network calculus-based timing analysis of AFDX networks with strict priority and TSN/BLS shapers, in: IEEE 13th International Symposium on Industrial Embedded Systems, 2018.

[24] A. Finzi, A. Mifdaoui, Worst-case timing analysis of AFDX networks with multiple TSN/BLS shapers, IEEE Access (2020).

[25] D. Maxim, Y.-Q. Song, Delay analysis of AVB traffic in time-sensitive networks (TSN), in: International Conference on Real-Time Networks and Systems, 2017.

[26] D. Thiele, R. Ernst, J. Diemer, Formal worst-case timing analysis of ethernet TSN's time-aware and peristaltic shaperss, in: Vehicular Networking Conference, 2015.

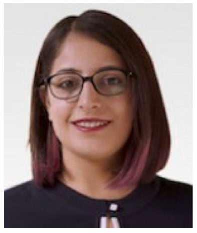
[27] L. Zhao, P. Pop, Z. Zheng, Q. Li, Timing analysis of AVB traffic in TSN networks using network calculus, in: Real-Time and Embedded Technology and Applications Symposium, 2018.

[28] L. Zhao, P. Pop, S. Craciunas, Worst-case latency analysis for IEEE 802.1Qbv time-sensitive networks using network calculus, IEEE Access (2018).

[29] M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, L. Lo Bello, Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support, Real-Time Syst. (2017).

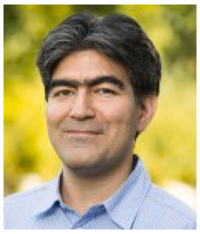
[30] G. Alderisi, G. Patti, L. Lo Bello, Introducing support for scheduled traffic over IEEE audio video bridging networks, in: IEEE International Conference on Emerging Technologies and Factory Automation, 2013.

[31] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, S. Afshar, S. Mubeen, Schedulability analysis of best-effort traffic in TSN networks, in: 26th IEEE International Conference on Emerging Technologies and Factory Automation, 2021.
[32] D. Thiele, R. Ernst, Formal worst-case performance analysis of time-sensitive ethernet with frame preemption, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation, 2016.
[33] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, M. Boyer, Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus, IEEE Trans. Ind. Electron. (2020).
[34] L. Lo Bello, M. Ashjaei, G. Patti, M. Behnam, Schedulability analysis of time-sensitive networks with scheduled traffic and preemption support, J. Parallel Distrib. Comput. (2020).
[35] L. Zhao, F. He, E. Li, J. Lu, Comparison of time-sensitive networking (TSN) and TTEthernet, in: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference, DASC, IEEE, 2018.
[36] M. Günzel, K. Chen, N. Ueter, G. von der Brüggen, M. Dürr, J. Chen, Timing analysis of asynchronized distributed cause-effect chains, in: 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium, RTAS, 2021.
[37] K. Gemlau, L. Köhler, R. Ernst, S. Quinton, System-level logical execution time: Augmenting the logical execution time paradigm for distributed real-time automotive software, ACM Trans. Cyber-Phys. Syst. (2021).
[38] H. Teper, M. Günzel, N. Ueter, G. von der Brüggen, J. Chen, End-to-end timing analysis in ROS2, in: 2022 IEEE Real-Time Systems Symposium, RTSS, 2022.
[39] M. Ashjaei, S. Mubeen, J. Lundbäck, M. Gålnander, K. Lundbäck, T. Nolte, Modeling and timing analysis of vehicle functions distributed over switched ethernet, in: 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017.
[40] S. Mubeen, M. Gålnander, J. Lundbäck, K. Lundbäck, Extracting timing models from component-based multi-criticality vehicular embedded systems, in: 15th International Conference on Information Technology : New Generations, 2018.
[41] M. Ashjaei, N. Khalilzad, S. Mubeen, M. Behnam, I. Sander, L. Almeida, T. Nolte, Designing end-to-end resource reservations in predictable distributed embedded systems, Real-Time Syst. (2017).
[42] J. Schlatow, M. Mostl, S. Tobuschat, T. Ishigooka, R. Ernst, Data-age analysis and optimisation for cause-effect chains in automotive control systems, in: 2018 IEEE 13th International Symposium on Industrial Embedded Systems, SIES, 2018.
[43] Jorge Martinez, Ignacio Sañudo, Marko Bertogna, Analytical characterization of end-to-end communication delays with logical execution time, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2018).
[44] Tobias Klaus, Florian Franzmann, Matthias Becker, Peter Ulbrich, Data propagation delay constraints in multi-rate systems: Deadlines vs. Job-level dependencies, in: Proceedings of the 26th International Conference on Real-Time Networks and Systems, RTNS '18, ACM, 2018.
[45] R. Bi, X. Liu, J. Ren, P. Wang, H. Lv, G. Tan, Efficient maximum data age analysis for cause-effect chains in automotive systems, in: Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022.
[46] Marco Dürr, Georg Von Der Brüggen, Kuan-Hsun Chen, Jian-Jia Chen, End-to-end timing analysis of sporadic cause-effect chains in distributed systems, ACM Trans. Embed. Comput. Syst. (2019).
[47] Tomasz Kloda, Antoine Bertout, Yves Sorel, Latency analysis for data chains of real-time periodic tasks, in: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2018.
[48] L. Köhler, P. Hertha, M. Beckert, A. Bendrick, R. Ernst, Robust cause-effect chains with bounded execution time and system-level logical execution time, ACM Trans. Embed. Comput. Syst. (2022).
[49] S. Mubeen, J. Mäki-Turja, M. Sjödin, Communications-oriented development of component-based vehicular distributed real-time embedded systems, J. Syst. Archit. (2014).
[50] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, K. Lundbäck, Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints, Int. J. Softw. Syst. Model. (2017).
[51] S. Mubeen, M. Gålnander, A. Bucaioni, J. Lundbäck, K. Lundbäck, Timing verification of component-based vehicle software with Rubus-ICE: End-user's experience, in: 2018 IEEE/ACM 1st International Workshop on Software Qualities and their Dependencies, IEEE, 2018.
[52] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, S. Mubeen, Synthesising schedules to improve QoS of best-effort traffic in TSN networks, in: 29th International Conference on Real-Time Networks and Systems, 2021.
[53] H. Lim, K. Weckemann, D. Herrscher, Performance study of an in-car switched ethernet network without prioritization, in: Communication Technologies for Vehicles, 2011.
[54] S. Kramer, D. Ziegenbein, A. Hamann, Real world automotive benchmarks for free, in: 6th Intl. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems, 2015.



Bahar Houtan received the bachelor degree in hardware engineering from Islamic Azad University (IAU), South Tehran Branch, Tehran, Iran, in 2014, and the M.Sc. degree in computer architecture from IAU, Science and Research, Tehran, in 2017. She is currently pursuing the Ph.D. degree in computer science with Mälardalen University, Västerås, Sweden. Her research interests include the IoT, blockchain technology, and networked real-time embedded systems.

Mohammad Ashjaei is a senior lecturer in the Complex Real-Time Systems (CORE) and the Heterogeneous Systems - Hardware Software Co- design (HERO) research groups at Mälardalen University in Sweden. Mohammad has received his Ph.D. degree in Computer Science in November 2016 from Mälardalen University. His main research interests include real-time systems, real-time distributed systems, scheduling algorithms on networks and processors, schedulability analysis techniques, resource reservation and reconfiguration mechanisms for real-time networks. He is also giving lectures on various topics related to embedded systems and data communication networks. He is a PC member and referee for several international conferences and journals, including IEEE Transactions on Industrial Informatics (TII), IEEE Transactions on Industrial Electronics (TIE), Elsevier's Journal of Systems Architecture, IEEE Transactions on Network and Services, Journal of Cloud Computing, and ACM Computing Surveys. He has organized and chaired several special sessions and workshops at the international conferences such as ETFA.



Masoud Daneshtalab is currently a Professor at Mälardalen University (MDU), and co-leading the Heterogeneous Systems research group. He is on the Euromicro board of directors, an editor of the MICPRO journal, and has published over 200 refereed papers. His research interests include HW/SW co-design, reliability, and deep learning acceleration.

Mikael Sjödin is since 2006 a professor of real- time system at Mälardalen University, Sweden. He is leading the research group Model-Based Engineering of Embedded Systems that focus on development of methods and tools for model-based engineering of embedded systems. Including: models for architectural and behavioral descriptions of system and requirements for systems, techniques for analyzing and transforming models, and runtime architectures for resource efficient, predictable embedded systems. Since 2012 he is the research director for Embedded Systems, a research environment with 200 active researchers at Mälardalen University. Between 2000 and 2006 Mikael held numerous industrial positions and worked as software architect and project manager. Mikael received his Ph.D. and M.Sc. in computer science from Uppsala University in 2000 and 1995 respectively.

Saad Mubeen is a Professor at Mälardalen University, Sweden. He has previously worked in the vehicle industry as a Senior Software Engineer at Arcticus Systems and as a Consultant for Volvo Construction Equipment, Sweden. He received his Ph.D. in Computer Science and Engineering from Mälardalen University Sweden in June 2014. He is a Senior Member of IEEE and a Co-chair of the Subcommittee on In-vehicle Embedded Systems within the IEEE IES Technical Committee on Factory Automation. His research focus is on model- and component-based development of predictable embedded software, modeling and timing analysis of in-vehicle communication, and end-to-end timing analysis of distributed embedded systems. Within this context, he has co-authored over 150 publications in peer-reviewed international journals, conferences and workshops. He has received several awards, including the IEEE Software Best Paper Award in 2017. He is a PC member and referee for several international conferences and journals respectively. He is a guest editor of Elsevier's Journal of Systems Architecture and Microprocessors and Microsystems, IEEE Transactions on Industrial Informatics (TII), ACM SIGBED Review, and Springer's Computing journal. He has organized and chaired several special sessions, tracks, and workshops at the international conferences such as IEEE's IECON, ICIT, ETFA, ISORC, DIVERSE, CRTS, to mention a few. For more information see http://www.es.mdh.se/staff/280-Saad_Mubeen.