Supporting End-to-end Data-propagation Delay Analysis for TSN Networks

Bahar Houtan[1] and Mohammad Ashjaei[1] and Masoud Daneshtalab[1] and Mikael Sjödin[1] and Saad Mubeen[1]

[1]Mälardalen University Sweden,
bahar.houtan@mdh.se, mohammad.ashjaei@mdh.se,
masoud.daneshtalab@mdh.se, mikael.sjodin@mdh.se,
saad.mubeen@mdh.se

Abstract. End-to-end data-propagation delay analysis allows verification of important timing constraints, such as age and reaction, that are often specified on chains of tasks and messages in real-time systems. We identify that the existing analysis does not support distributed task chains that include the Time-Sensitive Networking (TSN) messages. To this end, this paper extends the existing analysis to allow the end-to-end timing analysis of distributed task chains that include TSN messages. The extended analysis supports all types of traffic in TSN, including the Scheduled Traffic (ST), Audio Video Bridging (AVB), and Best-Effort (BE) traffic. Furthermore, the extended analysis accounts for the synchronization among the end stations that are connected via TSN. The applicability of the analysis is demonstrated using an automotive-application case study.

1 Introduction

Many embedded functions in modern vehicles are modeled as chains of tasks that can be distributed over multiple end stations (nodes) via a network. There are various types of timing constraints such as *Age* and *Reaction* that are often specified on these distributed chains. The age constraint constrains the age or freshness of the data from the input to the output of the chain. Whereas, the reaction constraint constrains the first reaction at the output of the chain corresponding to the data that arrived at the input of the chain[1]. These constraints are included in the timing model of the AUTOSAR standard [1] and have been translated to several modeling languages in the vehicular domain [2]. The developers of these functions are required to verify that these timing constraints are satisfied at the design time. These timing constraints can be verified by performing the end-to-end data-propagation delay analysis of these constrained distributed task chains [3–5]. Note that this type of analysis is applicable to both single node and distributed embedded systems. In the case of distributed embedded systems, there are several works that have incorporated various legacy

[1] The age and reaction constraints and the corresponding delays are discussed in detail in Section 3.

on-board real-time network protocols in this analysis, e.g., Controller Area Network (CAN) [6] and legacy Ethernet [7].

We identify that the existing end-to-end data-propagation delay analysis does not support the analysis of task chains that utilize the Time-Sensitive Networking (TSN) messages. TSN is a set of IEEE standards that are based on switched Ethernet. There are several works that provide response-time analysis for various types of traffic in TSN [8–10]. However, none of these analyses are integrated to the end-to-end data-propagation delay analysis. To this end, this paper proposes an end-to-end data-propagation delay analysis to allow the timing analysis of distributed task chains that encompass all types of TSN traffic classes, including Scheduled Traffic (ST), Audio Video Bridging (AVB), and Best-Effort (BE) traffic classes. The main contributions of the paper are as follows:

- We present a comprehensive system model for distributed embedded systems that use TSN for network communication. 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 develop an end-to-end data-propagation delay analysis of distributed task chains that include various types of TSN messages. The presented end-to-end timing analysis incorporates response-time analysis of TSN.
- We demonstrate the applicability of the proposed analysis on a vehicular use case.

2 Background and Related Work

2.1 Time-Sensitive Networking (TSN)

TSN standards are recently developed by the TSN task group in IEEE standardization. These 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 [11–13]. Among several features, the TSN standards allow temporal isolation of the ST traffic that is transmitted according to an offline schedule. Another feature is the Time-Aware Shaper (TAS) that can realize the temporal isolation using a set of gates controlling the transmission of traffic on a port of a TSN switch. The gates can stop the transmission of lower priority traffic in favour of the urgent ST traffic which in turn guarantees low-jitter transmission for the ST traffic. 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. The network can have multiple AVB classes, e.g., classes A and B, where class A has higher priority than class B, and they undergo the CBS mechanism for transmission. According to the CBS mechanism, a credit is configured per class of traffic on each port and the traffic associated to the class can only be sent 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.

2.2 Related Work

Several schedulability analysis techniques have been proposed in the literature to calculate the delays of traffic crossing through a TSN network in the worst-case scenarios. 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., [14] and the improved technique given in [15]. In addition, the work in [16] proposed a technique based on the trajectory approach to compute the delays of classes A and B. The technique obtains tighter bound compared to the previous techniques, e.g., compared to [14]. Later, the work in [17] proposed the notion of eligible interval that could provide a bound for delays per frame leading 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 [18] and [19] 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 [20] 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 [21] presented an analysis based on calculating the accumulation of delays, whereas the works in [22] and [23] used network calculus to check the schedulability of TSN messages. In addition, the technique in [24] presented a response-time analysis for classes A and B messages considering the CBS and support for the ST that is proposed in [25]. Furthermore, the work in [10] extends the technique in [24] for supporting the BE traffic. 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 the gate mechanism [26]. Therefore, the work in [27] proposed an analysis considering frame preemption under the IEEE 802.3br standard. Further, the work in [9] presented a technique that calculates 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 [8] 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.

In the context of 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 delays in the chains of tasks that include TSN messages. The existing end-to-end data-propagation delay analysis that computes these delays incorporates the analysis of various legacy real-time networks, such as CAN [5] and legacy Ethernet [28]. The work in [6] and [4] presented an end-to-end data-propagation delay analysis for automotive applications, where some of the techniques have been also implemented in tools to support component-based software development, e.g., [5, 7, 29, 30]. However, to the best of our knowledge, there is no existing work that incorporates the analysis of TSN into the end-to-end data-propagation delay analysis. This paper presents an end-to-end delay analysis considering data chains in TSN networks with all types of traffic classes, including ST, AVB and BE classes.

3 Data-propagation Delays

Real-time systems are often modelled with chains of tasks. In order 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 (age and reaction) should be calculated and compared with the corresponding age and reaction constraints. Consider a task chain consisting of three tasks τ_1, τ_2 and τ_3, as shown in Figure 1. The tasks in this chain 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. The tasks use register-based communication, i.e., they communicate to each other and to their environment by means of writing data to and reading 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 new data can be overwritten on the previous data in the register before the reader has read the previous data. The data read by τ_1 from Reg-1 corresponds to input of the chain. Similarly, the data written to Reg-3 by τ_3 corresponds to output of the chain.

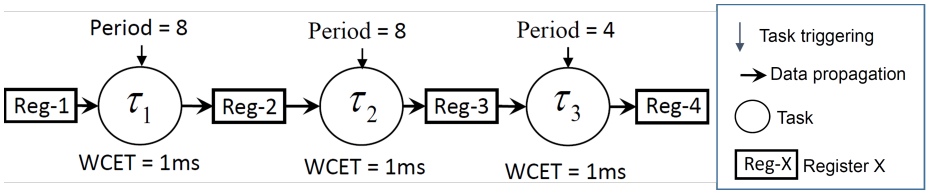

Fig. 1. An example of a task chain that uses register-based communication.

As the tasks are activated independently and some tasks have different periods, the data can traverse through the chain via multiple paths from the input to the output of the chain as shown in Figure 2. These paths are called timed paths. Due to multiple timed paths, there can be various delays that the data can experience from the input to the output of the chain. Two such delays that are common in the automotive systems are *age* and *reaction* delays [3]. The 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 age delay analysis, we are interested to identify the longest time difference between the input data and the last sample of corresponding output data. In addition, 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. Possible age and reaction delays for the chain in Figure 1 are shown in Figure 2. The age delay is important, in particular, for control applications where freshness of the data is of value. Whereas, the reaction delay is important in the applications where the first reaction to the input is of value.

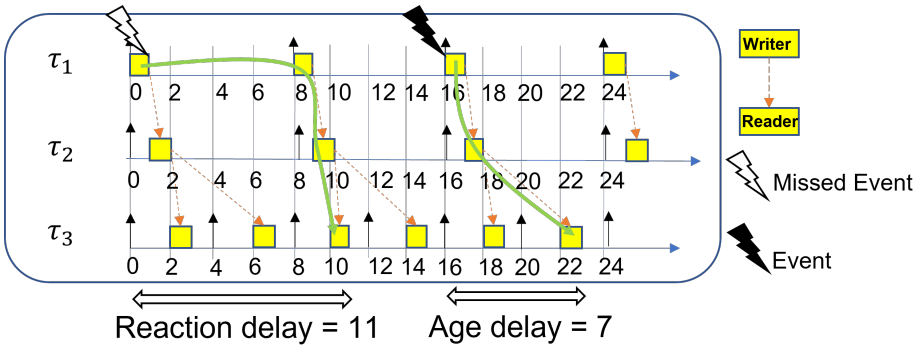

Fig. 2. Age and Reaction delays in the task chain depicted in Figure 1.

Although the data-propagation delays are discussed in the context of single-core processors, these delays are equally valid in distributed embedded systems. Let us consider a distributed task chain in a distributed embedded system depicted in Figure 3, where two nodes are connected via a network. In this example, the tasks are activated periodically with periods of 6ms and 3ms, 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 data path from the sender task to the receiver task. For example, when the network in not capable of initiating communication independent of the nodes, a message can only be queued for transmission at the network interface when the sending task sends it. This is the case of many network protocols, including CAN [31]. In this case, the message inherits period from its sender task. Furthermore, the data paths in a distributed task chain also depends upon if the network supports

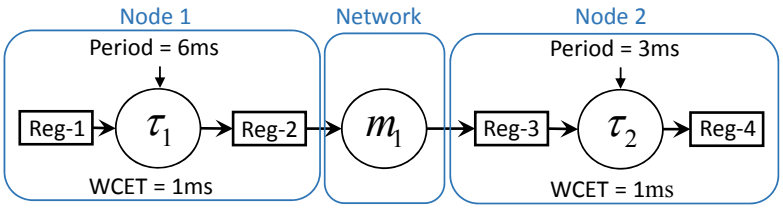

Fig. 3. A multi-rate chain in a distributed embedded system.

synchronization of nodes (e.g., Switched Ethernet) or not (e.g., CAN network). Figure 4 shows an execution trace when the nodes are synchronized in the system shown in Figure 3. The age and reaction delays in this distributed chain are also identified in Figure 4.

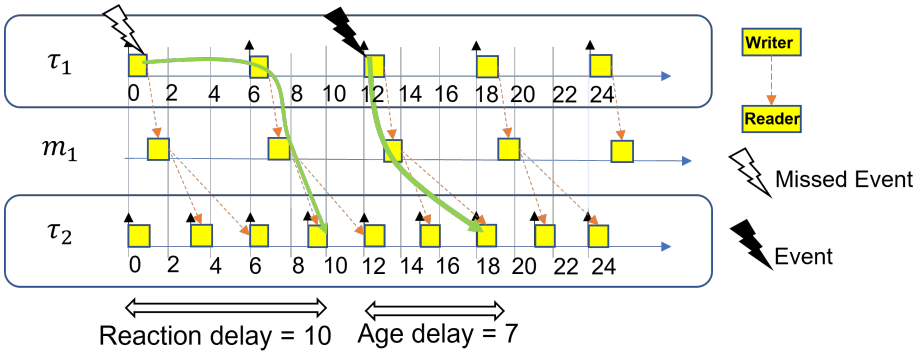

Fig. 4. A possible execution trace for the distributed embedded system example shown in Figure 3 when source and destination nodes are synchronized.

A possible execution trace of the distributed task chain in Figure 3 when the nodes are not synchronized is shown in Figure 5. To create worst case conditions when the nodes are not synchronized, we assume that the message receiving task τ_2 is activated "just before" the message is received at the receiver node. Hence, the current instance of τ_2 will miss the read access of the message. The message will be read by the next instance of τ_2 as shown in Figure 5. The corresponding age delay is also identified in this figure. To increase readability, we draw the same execution trace separately for the case of reaction delay in the distributed task chain (shown in Figure 3) when the nodes are not synchronized as depicted in Figure 6.

In the case of TSN, the messages belonging to some traffic classes (AVB and BE) can be activated by the sending tasks similar to CAN. Moreover, the messages in TSN can be activated independently by the network as is the case

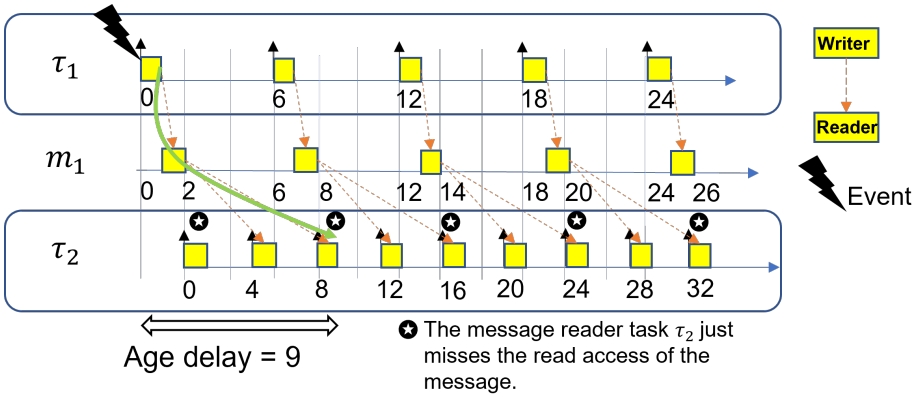

Fig. 5. A possible execution trace for the distributed embedded system example shown in Figure 3 when source and destination nodes are not synchronized (age delay).

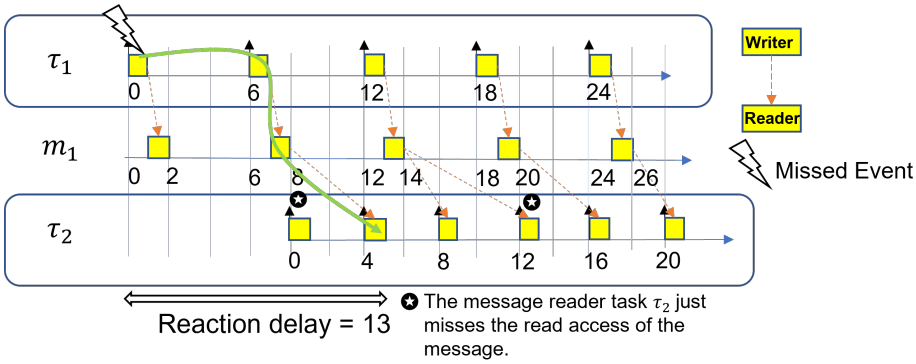

Fig. 6. A possible execution trace for the distributed embedded system example shown in Figure 3 when source and destination are not synchronized (reaction delay).

of ST traffic class. The messages belonging to the ST class are not supported by the data-path computation algorithm in the existing analysis. Hence, the existing end-to-end data-propagation delay analysis [3] requires extensions to support messages belonging to all TSN traffic classes. This paper extends the existing end-to-end data-propagation delay analysis by supporting distributed task chains that contain messages belonging to various types of traffic in TSN.

4 System model

In this section, we formally present the system model of a distributed system that consists of two or more end stations (single-core processors) that are connected by a TSN network. The system, denoted by $\mathcal{S}$, 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.

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

where the set of transactions and end stations are formally expressed subsequently by Eq. (2) and Eq. (3):

$$\Gamma := \{\Gamma_1, ..., \Gamma_n\} \tag{2}$$

$$\mathcal{E} := \{\mathcal{E}_1, ..., \mathcal{E}_n\} \tag{3}$$

4.1 End station model

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

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

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

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

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

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 n^{th} instance of the task τ_{ijk} 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}. Note that the n^{th} 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 and $\mathcal{L}$ 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 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.

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

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 A. In this model, the ST class has the highest priority, while AVB (classes A and B) has a lower priority than ST. However, class A has a higher priority than class B. Moreover, the BE class has the lowest priority among all the classes. The size of data (payload) 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 [32]. Therefore, the set $\mathcal{O}_{jk}$ for non-ST traffic 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 [10]. Moreover, the activation time of the n^{th} 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 period from the task from which they are transmitted. Therefore, T_{ijk} indicates the period of the k^{th} task (sending task) belonging to the i^{th} end station and being part of the j^{th} transaction. Moreover, we denote the n^{th} instance of the message m_{jk} by $m_{jk}(n)$.

4.4 Transaction

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.

Figure 7 shows an example of a TSN-based distributed embedded system with the model presented in this paper. There are two end stations connected to the network. More specifically, end station $\mathcal{E}_1$ and $\mathcal{E}_2$ are connected via links l_1 and l_2 to switch 1 ($SW1$). The system has two transactions as shown in Figure 7, namely Γ_1 and Γ_2. These transactions are further elaborated in Figure 8.

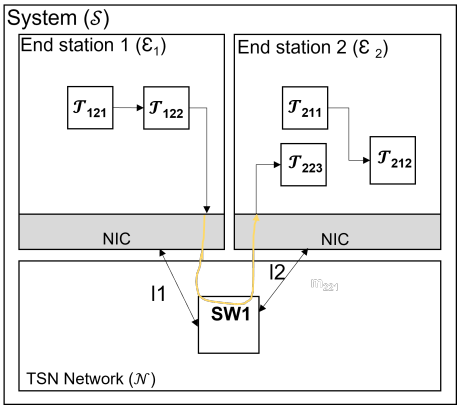

Fig. 7. Example of a distributed vehicular embedded system based on TSN.

A transaction that is within a single end station only includes tasks from this end station. Hence, the initiator and terminator end stations of the transaction are the same. For example, the transaction Γ_1 initiates and terminates inside the end station $\mathcal{E}_1$, as shown in Figure 8. On the other hand, transaction Γ_2 is distributed over two end stations.

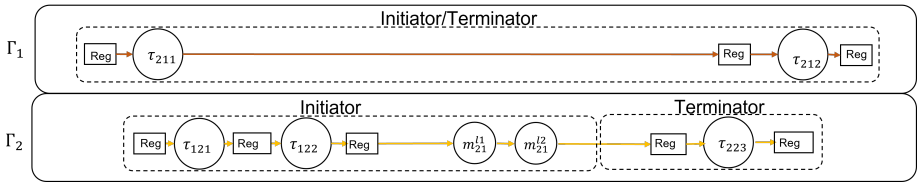

Fig. 8. Example of transactions.

Trigger modes According to [33], 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, 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. These modes represent the chains that are presented in Section 3. In this paper, we assume that a reader task is independently triggered and can receive the last available instance of the message from the network.

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 right after the execution of its source task is finished. For example, the messages assigned to non-ST classes in TSN networks (AVB or BE) can use the link as soon as the message's source task (the writer task) is executed, and if they obtain free bandwidth to use the link.

Transaction Constraints A constraint on transaction, denoted by Cr_j, defines the maximum allowed value of the delays, such as age (Age_j), reaction ($Reac_j$). Moreover, a deadline constraint (D_j) can be specified on a transaction. This constraint constrains the end-to-end response time of the transaction, which corresponds to response time of the last task in the transaction. These constraints are shown in Eq. (12) for transaction Γ_j:

$$Cr_j := \{Age_j, Reac_j, D_j\} \tag{12}$$

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 [3, 5] that considers legacy networks like CAN. The analysis requires computation of all relevant data paths (also called timed paths) within the distributed task chains.

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 is selected according to a set of Boolean functions [3].

The first condition in identifying a valid timed path is that a reader task (say τ_{dbe}) should not be activated before the writer task (say τ_{abc}). The condition defined in Eq. (13) is also known as activation time travel ($att()$), where a reader task is activated before the writer task. The activation time travel should not happen in valid timed paths. Note that $a(n)$ shows the activation time of the n^{th} instance of the task computed according to Eq. (6).

$$att(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) = \alpha_{dbe}(m) < \alpha_{abc}(n) \tag{13}$$

Where, Eq. (13) is true in case of the activation time travel between the writer and reader tasks. Otherwise, the condition is false, which is desirable.

Moreover, the writer and reader tasks shoud not overlap in a valid timed path. The critical function ($crit()$), shown in Eq. (14), returns true if the reader task τ_{abc} is activated before the writer task is completed. Otherwise, this function returns false, which is desirable. Note that R_{abc} is the worst-case response time of the writer task according to the system model presented earlier.

$$crit(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) = \alpha_{dbe}(m) < \alpha_{abc}(n) + R_{abc} \tag{14}$$

Where, Eq. (14) is true when the reader task is activated before the completion of the writer task. In such case, the reader task misses the data from the writer task.

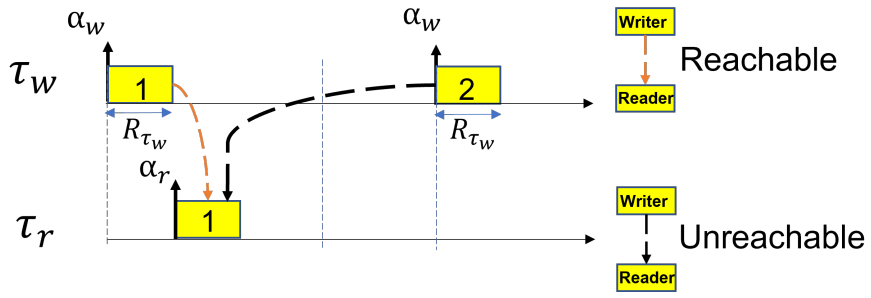

Fig. 9. End station tasks with different activation times.

Figure 9 shows an example of the case where the reader task is activated just after the execution of the writer task is completed. In such 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, which will be excluded by Eq. (13) and Eq. (14).

Moreover, the reader and writer task instances can only overlap when they are in the same end station; and if the priority of the reader task τ_{dbe} is less

than the priority of the writer task τ_{abc}. This is taken into account according to the wait function ($wait()$) in Eq. (15), where P indicates the priority of the task according to the system model.

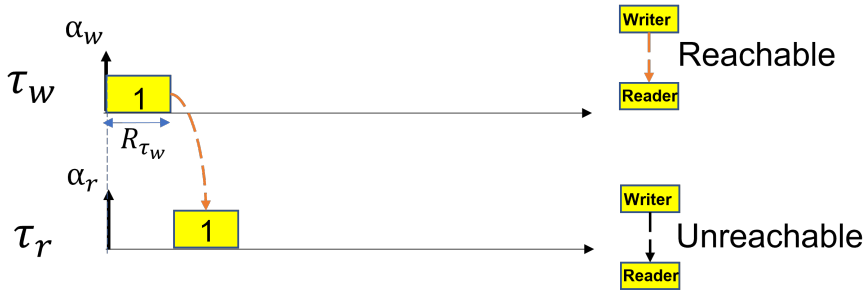

Fig. 10. End station tasks when activated at the same time.

$$wait(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) = P_{dbe} < P_{abc} \tag{15}$$

Figure 10 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 Figure 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 Figure 11. However, only one of those timed paths is a reachable timed path from the input to the output of the transaction.

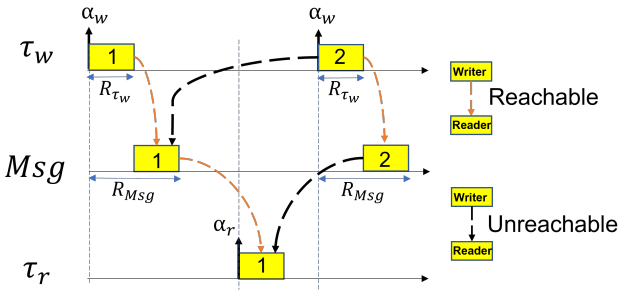

Fig. 11. Timed paths.

The forward reachability of two tasks in the timed path according to the aforementioned functions is examined based on the forward reachability function $forw()$ in Eq. (16).

$$\begin{aligned} &forw(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) = \\ &\neg att(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) \wedge \\ &(\neg crit(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) \vee wait(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m))) \end{aligned} \tag{16}$$

Further, 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 in the reader task will be overwritten. In order to detect this situation we should make sure that an instance of a writer task can reach to an instance of a reader task, while the next instance of the writer task cannot reach to the same reader task. This can be detected when the function in Eq. (17) returns true, where $\tau_{abc}(n+1)$ represents the next instance of the task instance $\tau_{abc}(n)$.

$$\begin{aligned} &reach(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) = forw(\tau_{abc}(n) \longrightarrow \tau_{dbe}(m)) \\ &\wedge \neg forw(\tau_{abc}(n+1) \longrightarrow \tau_{dbe}(m)) \end{aligned} \tag{17}$$

After checking the reachability between two task instances, we can 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. (18) where a timed path tp_j^i belongs to the transaction Γ_j. Moreover, for simplicity of the equation, two consecutive task instances in the timed path are shown by τ_w and τ_r.

$$reach(tp_j^i) = \prod reach(\tau_w \longrightarrow \tau_r) \tag{18}$$

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. (19) assuming that there are z timed paths in the transaction.

$$TP_j^{reach} = \{reach(tp_j^i); i = 1..z\} \tag{19}$$

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. 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. Accordingly, the reachability of an ST message's instance to a reader task instance 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 also take into account the activation times of the ST message per link in the route of the message. An example of a transaction is shown in Figure 12, in which the ST class is utilized for the communication between two end stations. 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 and the second task 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.

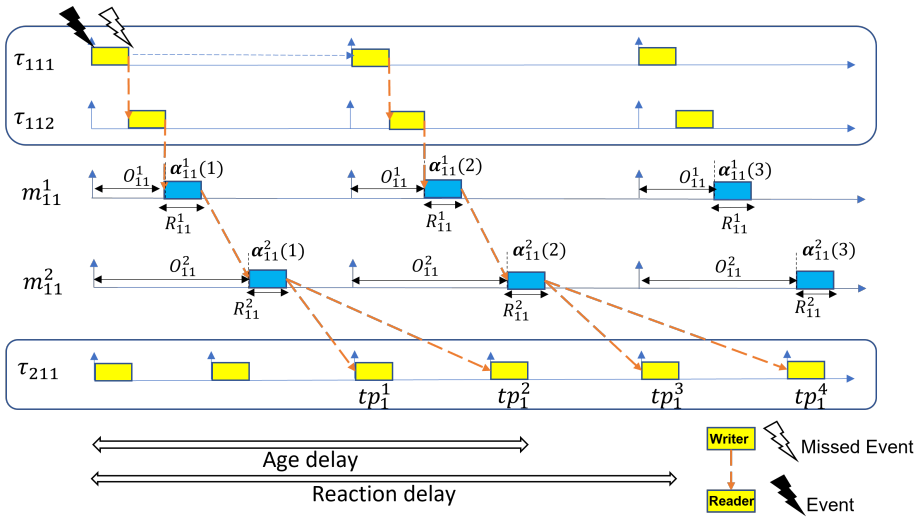

Fig. 12. Timed path with ST messages.

As it can be seen in Figure 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 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.*

Proof. ST offsets at the subsequent links in the route of the message are defined in a way to satisfy a set of constraints as presented in [32], such as, constraints on

the frame size, overlapping of messages on a link, order of traversed links, and deadline satisfaction. Given that the ST offsets satisfy all the aforementioned constraints, then it is guaranteed that the arrival time of the message is always before its deadline. In addition, given the offline schedule from [32] it is guaranteed that the offsets are increasing over the links on the path of the message, i.e., there will not be backward timed path over several links for an ST message. Another important aspect is 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 instance of the message. Thus, there is always one path for an ST message over several links. Therefore, it is enough to consider the activation of the ST message on the last link when extracting the reachable timed paths.

Based on Lemma 1 in case of the scenario depicted in Figure 12, the message at its last hop (m_{11}^2) is enough to consider in the identification of reachable timed paths for the end-to-end data-propagation delay analysis, which is shown in Figure 13. Hence, the existing analysis needs to be extended in order to support the TSN classes.

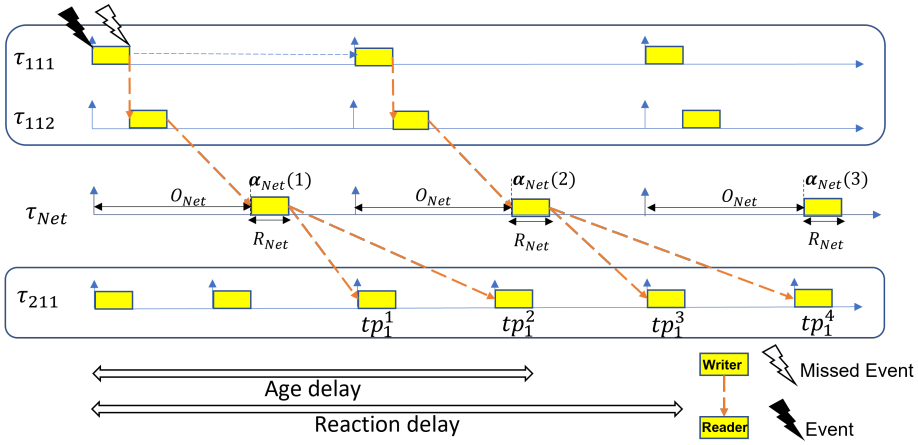

Fig. 13. Modelling the ST message at the last hop.

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 it as a separate task which simplifies the incorporation of the ST message in the existing analysis. Eq. (20) shows the model of such a task. We regard this task as the network task and denote it by τ_{Net}. This task corresponds to the message m_{jk}^r, where r is the last link connected to the receiving end station. We assume that there will be one such task per ST message.

$$\tau_{Net} := \langle P_{Net} = P_{jk}, C_{Net} = TT(Size_{jk}), T_{Net} = T_{jk}, J_{Net} = J_{jk}, O_{Net} = O^r_{jk} \rangle \tag{20}$$

where, $TT()$ calculates the transmission time of the message based on the size of the message and the network total bandwidth. Modeling of the ST message with a task allows us to use Eq. (19) to evaluate the reachability of timed paths corresponding to a transaction, where τ_{Net} can be a reader or writer task in a transaction, while its activation times is computed according to Eq. (10).

5.3 Accounting for the non-ST messages

In case of non-ST messages that are scheduled without offsets, i.e, AVB and BE classes, the approach to compute reachable timed paths in the existing end-to-end data-propagation delay analysis [3] holds good. 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 Figure 14, the non-ST message m_{11} has no offset in the set of two links in its route to the reader task, namely τ_{211}. As a result, the activation time of the message is assumed to be the same as its predecessor writer task. In Figure 14, each instance of m_{11} is not transmitted immediately after the completion of the sender task τ_{112} because we assume the message received interference from other higher priority messages. By knowing the worst-case response time of the message (R_{11}) the reachable instance of the reader task can be determined. For example, consider tp^1_1 and tp^2_1 in Figure 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. (16) is used to evaluate the reachability of timed paths corresponding to a transaction.

5.4 Worst-case delay analysis

The age and reaction delays are derived based on timed paths. According to [3] the age delay for a timed path tp^n_j belonging to the transaction Γ_j is calculated by Eq. (21), which calculates the time difference between the input data and the last sample of the corresponding output data.

$$\Delta_{age}(tp^n_j) = \alpha_{last}(tp^n_j) + RT_{last}(tp^n_j) - \alpha_{first}(tp^n_j) \tag{21}$$

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. Moreover, $\alpha_{last}()$ 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 delay is calculated by Eq. (22), where $Pred()$ represents the first instance of the timed path before the timed path under analysis tp^n. Note

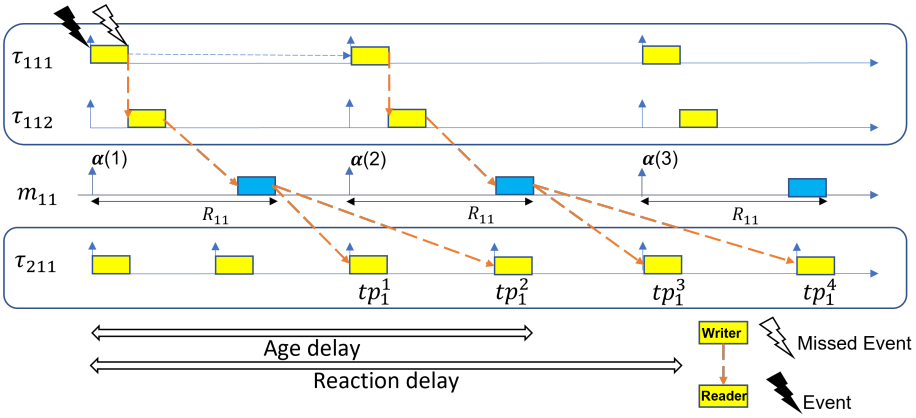

Fig. 14. Timed path with non-ST messages.

that the effect of just missing an event at the input of the task chain is covered by $Pred()$.

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

The age and reaction delays for all timed paths for every transaction should be extracted and the longest corresponding values represent the worst-case age and reaction delays, which are calculated according to Eq. (23).

$$\begin{aligned} \Delta_{age}(\Gamma_j) &= \{max(\Delta_{age}(tp_j^n)); n = 1...z\} \\ \Delta_{reac}(\Gamma_j) &= \{max(\Delta_{reac}(tp_j^n)); n = 1...z\} \end{aligned} \tag{23}$$

6 Vehicular Application Case Study

In this section, we discuss an vehicular use case that is used to evaluate the presented end-to-end data-propagation delay analysis.

6.1 Evaluation settings

We consider a use case that consists of 14 end stations (nodes) that are connected by a two-switch TSN network as illustrated in Figure 15. For the purpose of the analysis, we assume each end station includes multiple tasks. 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. Each transaction initiates and terminates by a task within different end stations. Each transaction includes four tasks in total. Besides, it includes two tasks per end station which participates in the transaction. In the source end station, the first task is a computation task and

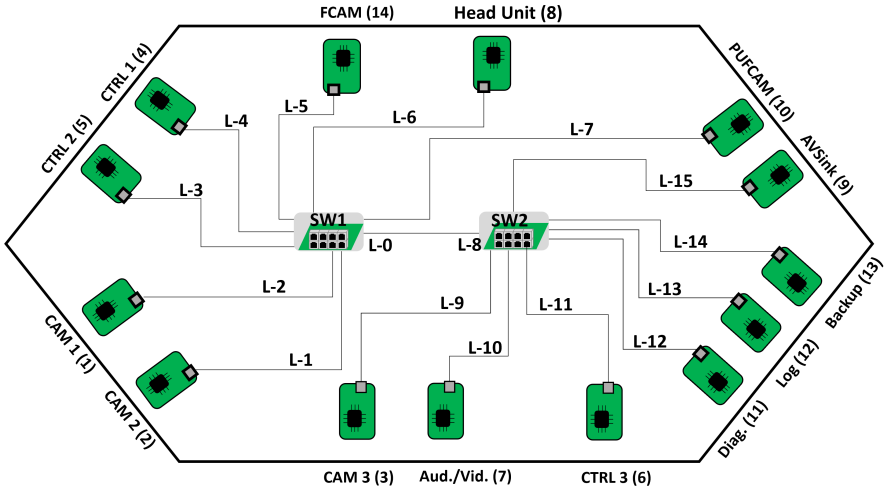

Fig. 15. Vehicular application use case topology.

the subsequent task is a communication task, which receives data as input, then prepares and injects messages to the network. In the destination end station of the transaction, 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 in [34]. The size of all messages are fixed to 1542 *Bytes* as the maximum Ethernet frame size and the execution time of each task is considered to be 0.5 *ms*. In each transaction, we assume that the priority of of each precedent task is higher than the priority of its subsequent task within an end station. Furthermore, we do not consider forking and joining of tasks in a transaction.

Table 1. The evaluation settings for the use-case based on distributed data chain.

Γ_j	Source ($\mathcal{E}_i$)	Source tasks (τ_{ijk}): [id, P_{ijk},C_{ijk}, T_{ijk}]		Message (m_{jk}): [id, P_{jk},$Size_{jk}$, T_{jk}, $\mathcal{O}^r_{jk}$]	Destination ($\mathcal{E}_i$)	Destination tasks (τ_{ijk}): [id, P_{ijk},C_{ijk}, T_{ijk}]	
		Computation	Communication			Communication	Computation
1	CAM1 (1)	[$\tau_{1,1,1}$,4,0.5,20]	[$\tau_{1,1,2}$,3,0.5,20]	[$m_{1,1}$, ST,1542,20,0.039]	HU (8)	[$\tau_{8,1,1}$,10,0.5,10]	[$\tau_{8,1,2}$,9,0.5,10]
2	CAM3 (3)	[$\tau_{3,2,1}$,4,0.5,20]	[$\tau_{3,2,2}$,3,0.5,20]	[$m_{2,2}$, ST,1542,20,0.012]	HU (8)	[$\tau_{8,2,3}$,8,0.5,10]	[$\tau_{8,2,4}$,7,0.5,10]
3	CTRL1 (4)	[$\tau_{4,3,1}$,4,0.5,10]	[$\tau_{4,3,2}$,3,0.5,10]	[$m_{3,3}$, ST,1542,10,0.065]	HU (8)	[$\tau_{8,3,5}$,6,0.5,10]	[$\tau_{8,3,6}$,5,0.5,10]
4	CTRL3 (6)	[$\tau_{6,4,1}$,4,0.5,10]	[$\tau_{6,4,2}$,3,0.5,10]	[$m_{4,4}$, ST,1542,10,0.078]	HU (8)	[$\tau_{8,4,7}$,4,0.5,10]	[$\tau_{8,4,8}$,3,0.5,10]
5	CTRL2 (5)	[$\tau_{5,5,1}$,2,0.5,10]	[$\tau_{5,5,2}$,1,0.5,10]	[$m_{5,5}$, ST,1542,10,0.026]	AVSink (9)	[$\tau_{9,5,1}$,10,0.5,10]	[$\tau_{9,5,2}$,9,0.5,10]
6	AV (7)	[$\tau_{7,6,1}$,6,0.5,10]	[$\tau_{7,6,2}$,5,0.5,10]	[$m_{6,6}$, A,1542,10,0]	AVSink (9)	[$\tau_{9,6,3}$,8,0.5,10]	[$\tau_{9,6,4}$,7,0.5,10]
7	AV (7)	[$\tau_{7,7,3}$,4,0.5,10]	[$\tau_{7,7,4}$,3,0.5,10]	[$m_{7,7}$, A,1542,10,0]	AVSink (9)	[$\tau_{9,7,5}$,6,0.5,10]	[$\tau_{9,7,6}$,5,0.5,10]
8	AV (7)	[$\tau_{7,8,5}$,2,0.5,10]	[$\tau_{7,8,6}$,1,0.5,10]	[$m_{8,8}$, A,1542,10,0]	AVSink (9)	[$\tau_{9,8,7}$,4,0.5,10]	[$\tau_{9,8,8}$,3,0.5,10]
9	CAM1 (1)	[$\tau_{1,9,3}$,2,0.5,20]	[$\tau_{1,9,4}$,1,0.5,20]	[$m_{9,9}$, B,1542,20,0]	PUFCam (10)	[$\tau_{10,9,1}$,8,0.5,10]	[$\tau_{10,9,2}$,7,0.5,10]
10	CAM2 (2)	[$\tau_{2,10,1}$,4,0.5,20]	[$\tau_{2,10,2}$,3,0.5,20]	[$m_{10,10}$, B,1542,20,0]	PUFCam (10)	[$\tau_{10,10,3}$,6,0.5,10]	[$\tau_{10,10,4}$,5,0.5,10]
11	CAM2 (2)	[$\tau_{2,11,3}$,2,0.5,20]	[$\tau_{2,11,4}$,1,0.5,20]	[$m_{11,11}$, B,1542,20,0]	PUFCam (10)	[$\tau_{10,11,5}$,4,0.5,10]	[$\tau_{10,11,6}$,3,0.5,10]
12	CAM3 (3)	[$\tau_{3,12,3}$,2,0.5,20]	[$\tau_{3,12,4}$,1,0.5,20]	[$m_{12,12}$, BE,1542,20,0]	HU (8)	[$\tau_{8,12,9}$,2,0.5,10]	[$\tau_{8,12,10}$,1,0.5,10]
13	CTRL1 (4)	[$\tau_{4,13,3}$,2,0.5,10]	[$\tau_{4,13,4}$,1,0.5,10]	[$m_{13,13}$, BE,1542,10,0]	PUFCam (10)	[$\tau_{10,13,7}$,2,0.5,10]	[$\tau_{10,13,8}$,1,0.5,10]
14	CTRL3 (6)	[$\tau_{6,14,3}$,2,0.5,10]	[$\tau_{6,14,4}$,1,0.5,10]	[$m_{14,14}$, BE,1542,10,0]	AVSink (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	-	-

Among the transactions, five transactions use ST 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 bandwidth is set to 1 $Gbps$. We set the idle slope 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. which are not presented in Table 2.

Finally, the age and reaction constraints specified on each transaction are depicted in Table 3.

Table 3. Timing constraints for all transactions

	Reac_j (ms)	Age_j (ms)
Cr_j	35	25

6.2 Evaluation setup and analysis results

We implemented the proposed analysis as an in-house tool. The configuration and message set in the vehicular application use-case were given as input to the implemented analysis. Table 4 shows the calculated age and reaction delays for each individual transaction. By comparing the calculated delays with the corresponding constraints specified on each transaction, we see that all the transactions meet their age and reaction constraints.

7 Conclusions and Future Works

In this paper, we identified that the existing end-to-end data-propagation delay analysis for distributed embedded systems does not support all traffic classes in TSN networks. In particular, the ST class that is scheduled offline is not supported. We proposed extensions to the existing analysis that now supports all

Table 4. Calculated age and reaction delays for each transaction.

Trans. (Γ_j)	Reaction Delay (ms)	Age Delay (ms)
1	31	21
2	32	22
3	23	13
4	24	14
5	21	11
6	22	12
7	23	13
8	24	14
9	31	21
10	32	22
11	33	23
12	35	25
13	24	14
14	25	15

traffic classes in TSN. The extensions are proposed such that the extended analysis is backward compatible with the legacy networks like CAN. We evaluated the presented analysis on a vehicular application use case. A potential future work is to integrate the proposed analysis with model-based software development frameworks for distributed embedded systems, e.g., Rubus-ICE [29].

Acknowledgements

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.

References

1. AUTOSAR Consortium, AUTOSAR Techincal Overview [online], Release 4.1, Rev.2, Ver.1.1.0., http://autosar.org.
2. Saad Mubeen, Thomas Nolte, Mikael Sjödin, John Lundbäck, and Kurt-Lennart Lundbäck. Supporting Timing Analysis of Vehicular Embedded Systems through the Refinement of Timing Constraints. *International Journal on Software and Systems Modeling*, January 2017.
3. N. Feiertag, K. Richter, J. Nordlander, and 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*, December 2008.

4. Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. End-to-end Timing Analysis of Cause-Effect Chains in Automotive Embedded Systems. *Journal of Systems Architecture*, October 2017.
5. Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. Support for End-to-End Response-time and Delay Analysis in the Industrial Tool Suite: Issues, Experiences and a Case Study. ComSIS Consortium, January 2013.
6. Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. Communications-oriented Development of Component-based Vehicular Distributed Real-time Embedded Systems. *Journal of Systems Architecture*, January 2014.
7. Mohammad Ashjaei, Saad Mubeen, John Lundbäck, Mattias Gålnander, Kurt-Lennart Lundbäck, and Thomas Nolte. Modeling and Timing Analysis of Vehicle Functions Distributed over Switched Ethernet. In *43rd Annual Conference of the IEEE Industrial Electronics Society*, October 2017.
8. Lucia Lo Bello, Mohammad Ashjaei, Gaetano Patti, and Moris Behnam. Schedulability Analysis of Time-Sensitive Networks with Scheduled Traffic and Preemption Support. *Journal of Parallel and Distributed Computing*, October 2020.
9. Luxi Zhao, P. Pop, Zhong Zheng, Hugo Daigmorte, and M. Boyer. Latency Analysis of Multiple Classes of AVB Traffic in TSN with Standard Credit Behavior using Network Calculus. *IEEE Transactions on Industrial Electronics*, September 2020.
10. Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, Sara Afshar, and Saad Mubeen. Schedulability Analysis of Best-effort Traffic in TSN Networks. In *26th IEEE International Conference on Emerging Technologies and Factory Automation*, September 2021.
11. L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara. Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems. *IEEE Transactions on Industrial Informatics*, November 2019.
12. L. Lo Bello and W. Steiner. A Perspective on IEEE Time-sensitive Networking for Industrial Communication and Automation Systems. *Proceedings of the IEEE*, April 2019.
13. Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. Time-sensitive Networking in Automotive Embedded Systems: State-of-the-Art and Research Opportunities. *Journal of Systems Architecture, 2021*, September 2021.
14. J. Diemer, D. Thiele, and R. Ernst. Formal Worst-case Timing Analysis of Ethernet Topologies with Strict-priority and AVB Switching. In *International Symposium on Industrial Embedded Systems*, June 2012.
15. Unmesh D. Bordoloi, Amir Aminifar, Petru Eles, and Zebo Peng. Schedulability Analysis of Ethernet AVB Switches. In *International Conference on Embedded and Real-Time Computing Systems and Applications*, August 2014.
16. Xiaoting Li and Laurent George. Deterministic Delay Analysis of AVB Switched Ethernet Networks Using an Extended Trajectory Approach. *Real-Time Systems*, January 2017.
17. J. Cao, P. J. L. Cuijpers, R. J. Bril, and J. J. Lukkien. Independent yet Tight WCRT Analysis for Individual Priority Classes in Ethernet AVB. In *International Conference on Real-Time Networks and Systems*, October 2016.
18. A. Finzi, A. Mifdaoui, F. Frances, and 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*, June 2018.
19. A. Finzi and A. Mifdaoui. Worst-case Timing Analysis of AFDX Networks with Multiple TSN/BLS Shapers. *IEEE Access*, June 2020.

20. D. Maxim and Y.-Q. Song. Delay Analysis of AVB traffic in Time-sensitive Networks (TSN). In *International Conference on Real-time Networks and Systems*, October 2017.
21. D. Thiele, R. Ernst, and J. Diemer. Formal Worst-case Timing Analysis of Ethernet TSN's Time-aware and Peristaltic Shaperss. In *Vehicular Networking Conference*, December 2015.
22. L. Zhao, P. Pop, Z. Zheng, and Q. Li. Timing Analysis of AVB Traffic in TSN Networks using Network Calculus. In *Real-Time and Embedded Technology and Applications Symposium*, August 2018.
23. L. Zhao, P. Pop, and S. Craciunas. Worst-case Latency Analysis for IEEE 802.1Qbv Time-sensitive Networks using Network Calculus. *IEEE Access*, July 2018.
24. Mohammad Ashjaei, Gaetano Patti, Moris Behnam, Thomas Nolte, Giuliana Alderisi, and Lucia Lo Bello. Schedulability Analysis of Ethernet Audio Video Bridging Networks with Scheduled Traffic Support. *Real-Time Systems*, July 2017.
25. G. Alderisi, G. Patti, and L. Lo Bello. Introducing Support for Scheduled traffic over IEEE Audio Video Bridging Networks. In *IEEE International Conference on Emerging Technologies and Factory Automation*, September 2013.
26. Mohammad Ashjaei, Mikael Sjödin, and Saad Mubeen. A Novel Frame Preemption Model in TSN Networks. *Journal of Systems Architecture*, June 2021.
27. D. Thiele and 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*, September 2016.
28. Mohammad Ashjaei, Nima Khalilzad, Saad Mubeen, Moris Behnam, Ingo Sander, Luís Almeida, and Thomas Nolte. Designing End-to-end Resource Reservations in Predictable Distributed Embedded Systems. *Real-time Systems*, June 2017.
29. Saad Mubeen, Harold Lawson, John Lundbäck, Mattias Gålnander, and Kurt-Lennart 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, May 2017.
30. Saad Mubeen, Mattias Gålnander, John Lundbäck, and Kurt-Lennart Lundbäck. Extracting Timing Models from Component-based Multi-criticality Vehicular Embedded Systems. In *15th International Conference on Information Technology : New Generations*, April 2018.
31. ISO 11898-1. Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High-speed Communication, ISO Standard-11898, Nov. 1993.
32. Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, and Saad Mubeen. Synthesising Schedules to Improve QoS of Best-effort Traffic in TSN Networks. In *29th International Conference on Real-time Networks and Systems*, April 2021.
33. Saad Mubeen, Mattias Gålnander, Alessio Bucaioni, John Lundbäck, and Kurt-Lennart 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, May 2018.
34. S. Kramer, D. Ziegenbein, and A. Hamann. Real World Automotive Benchmarks for Free. In *6th Intl. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems*, July 2015.