Holistic Modeling of Time Sensitive Networking in Component-based Vehicular Embedded Systems

Saad Mubeen, Mohammad Ashjaei, Mikael Sjödin
Mälardalen University, Sweden
{saad.mubeen, mohammad.ashjaei, mikael.sjodin}@mdh.se

***Abstract*—This paper presents the first holistic modeling approach for Time-Sensitive Networking (TSN) communication that integrates into a model- and component-based software development framework for distributed embedded systems. Based on these new models, we also present an end-to-end timing model for TSN-interconnected distributed embedded systems. Our approach is expressive enough to model the timing information of TSN and the timing behaviour of software that communicates over TSN, hence allowing end-to-end timing analysis. A proof of concept for the proposed approach is provided by implementing it for a component model and tool suite used in the vehicle industry. Moreover, a use case from the vehicle industry is modeled and analyzed with the proposed approach to demonstrate its usability.**

I. Introduction

The advanced features in modern vehicles can be largely attributed to the innovation in computer controlled functionality [1]. Much of this functionality requires new levels of computational power, e.g., due to data-intensive sensors such as video camera, radar and ultrasonic sensors. Moreover, complex coordination among the subsystems and high-bandwidth communication among the compute units, also called the Electronic Control Units (ECUs), is needed to realize the advanced functionality. These requirements are expected to increase further based on the recent trends in the vehicle industry [2], [3]. In order to meet these requirements, architectures based on powerful ECUs that are connected by high-bandwidth low-latency real-time communication networks emerge [4], [5].

There is a wealth of existing works [3], [6], [7], [8] that aim at providing model- and component-based software development techniques for vehicular embedded systems that are deployed on powerful ECUs, which are connected by low-bandwidth and low-latency real-time communication networks, such as Controller Area Network (CAN) [9]. There is also emerging work to support the development of vehicular embedded systems that use high-bandwidth on-board real-time communication [10], [11], [12], [13]. One particular example is Audio/Video Bridging (AVB) [14], which is a set of IEEE standards supporting high-bandwidth (up to 100 Mbit/s) on-board real-time communication. However, AVB does not support low-latency communication that is required by time-sensitive control messages.

In a recent effort to support high-bandwidth, low-latency and real-time on-board network communication, the IEEE Time-Sensitive Networking (TSN) task group has developed a set of standards [15], [16], [17]. The solutions based on these standards are becoming attractive in various domains due to the features that support different classes of traffic, including real-time, low-latency, time-triggered and non-real-time traffic, resource reservation for the traffic classes, clock synchronization and traffic shaping. While there are several works to provide configuration and design optimization [18], [19], [20], timing analysis for some specific features [21], [22], [23], [24] and hardware support [25] for TSN, there is no existing component model, modeling language or approach to support the modeling of TSN. In order to fully utilize the potential of the TSN technology, a new approach is needed to support the modeling of TSN in the software architectures of distributed embedded systems. The modeling approach should be expressive enough to model timing properties and requirements of TSN on the software architectures to support timing analysis. Moreover, the approach should be interoperable to allow its integration to the existing model- and component-based software development frameworks and component models. We envision such a modeling approach for TSN to be instrumental for the vehicle industry in developing the vehicular applications that utilize TSN as the backbone for on-board network communication.

As the main contribution of this paper we present the first modeling approach for TSN that allows integration in a software development environment and allows to analyze end-to-end delays in distributed embedded systems interconnected with TSN. The proposed approach is integrated to the existing model- and component-based software development framework for vehicular distributed embedded systems. The approach is expressive enough to allow specification of timing information on the software architectures of these systems. In this regard, the paper presents a comprehensive end-to-end timing model augmented by TSN. As a proof of concept, the proposed approach is implemented in the Rubus Component Model (RCM) [26] and Rubus-ICE tool suite, which have been used in the vehicle industry for developing predictable embedded systems for 25 years [27]. The validity of the approach is demonstrated by modeling and analyzing an automotive-application use case.

II. Background and Related Approaches

A. Time Sensitive Networking (TSN)

This paper focuses on two standards in the set of TSN standards, namely IEEE 802.1Qbv-2015 [15] and IEEE 802.1Qbu-2016 [16]. These standards include enhancements to support scheduled traffic and preemption of messages respectively. The

two standards, which have been included in the IEEE 802.1Q-2018 standard [28], are very suitable for industrial networks, where the scheduled traffic and real-time flows have to coexist on the same network with best-effort transmissions. Moreover, a *credit-based shaper* (CBS) algorithm is defined to prevent the transmission of traffic bursts. This algorithm applies to only stream reservation classes, namely A and B. Each such class has an associate *credit* parameter. Any pending message in the queue of a traffic class can only be transmitted if the corresponding credit is zero or higher. During the transmission, the credit is consumed at a constant rate, known as the *sendSlope*. The credit is replenished with a constant rate, called the *idleSlope*, in two cases: (i) when the messages in the queue are pending for transmission, and (ii) when there are no more pending messages in the queue, but the credit is negative. If there is no message in the queue and the credit is positive then the credit is immediately reset to zero.

The IEEE 802.1Qbv-2015 enhancements provide temporal isolation for the scheduled traffic. In particular, *transmission gates* are associated with each queue of a switch port, and transmission from a queue is only allowed if the relevant gate is open. The gates operation follows a cyclic table predefined by the network designer. *Preemption support* in the IEEE 802.1Qbu-2016 standard introduces *express* and *preemptable* traffic classes. Express traffic can preempt the preemptable traffic, but it cannot be preempted itself. The preemptable traffic cannot preempt other classes. Preemption support can be combined with the CBS and the gate mechanisms. For instance, the scheduled traffic queue can be set to express, while all the other queues are preemptable.

B. Related Approaches, Languages and Component Models

The component models and modeling languages that are based on the principles of Model-Based Engineering (MBE) [29] and Component-Based Software Engineering (CBSE) [30] are proving effective in dealing with the complexity of embedded software in vehicular systems. It is estimated that up to 90% of the software can be reused from the previous releases of the vehicle or other in-house projects by applying MBE and CBSE [31]. There are several component models in the vehicular domain that support modeling of onboard real-time network communication, such as AUTOSAR [32], RCM [26], ProCom [33], COMDES [34], CORBA [35], just to name a few. A majority of existing component models allow modeling of low-bandwidth real-time onboard networks, e.g., CAN. There are very few works such as [13] that support modeling of high-bandwidth onboard networks that are based on Ethernet; however, only in the academic settings. AUTOSAR, complemented by the SymTA/S tool[1] [36] facilitate modeling of Ethernet, but the support for modeling TSN is still missing. Moreover, being a commercial tool, SymTA/S does not expose any details about the underlying techniques. In [37], a work in progress is discussed for developing an approach to configure TSN network and verify its configuration. To the best of our knowledge, there is no existing work that supports modeling of TSN-based communication in the software architectures of component-based distributed embedded systems.

In this paper, we choose to use RCM for the proof-of-concept implementation of the proposed modeling approach. The reasons for selecting RCM include support for explicitly modeling timing information, pipe-and-filter communication (most commonly used style supporting interoperability), run-to-completion semantics of software components (compliant with various standards like AUTOSAR), aggressively resource efficient and extremely low run-time and memory footprints. RCM is complemented by the Rubus real-time operating system (RTOS), which is a certified RTOS (according to ISO 262626), and the Rubus-ICE tool suite that consists of modeling tools, code generators, analysis tools and run-time infrastructure. Despite being a commercial component model, RCM and its tool suite disclose the underlying scientific techniques and analysis methods and make them transparent to the scientific community [26], [38], [27], [39]. Whereas, these details are not accessible to the scientific community in the case of other commercial component models and tools in the vehicular domain [40].

III. End-to-end Timing Model Augmented by TSN

This section first discusses the end-to-end timing model. Thereafter, the section identifies the missing information from the model to support TSN and presents augmentation of TSN to the end-to-end timing model.

A. End-to-end Timing Model

The end-to-end timing model comprises timing properties, timing requirements and dependencies of all elements in the nodes and networks within a distributed embedded system. Such a model is required by the analysis engines to perform the end-to-end timing analysis [41], [42] as shown in Fig. 1. The end-to-end timing model is composed of four models: node timing model, network timing model, system linking model and timing requirements model as shown in Fig. 1(b). A distributed embedded system $\mathcal{S}$ consists of two or more nodes ($\mathcal{E}_i$) and one or more networks ($\mathcal{N}_i$). The number of nodes and networks in the system is represented by $|\mathcal{E}|$ and $|\mathcal{N}|$ respectively. In this paper we consider only single-core nodes for the sake of simplicity and keeping the discussion focused on the network communication based on TSN. We refer the reader to [43] for the details of multi-core node model.

*1) **Node Timing Model**:* The software architecture in a node consists of software components (SWCs) and their interactions as shown in Fig. 1(a). A SWC is a design-time entity, which may correspond to an operating system task at run-time. We consider a one-to-one mapping between a SWC and a task, which is often the case in many academic and industrial component models and supporting RTOSs such as Rubus, ProCom and COMDES. The node timing model is based on the transactional task model [44], i.e., each node consists of one or more transactions Γ_{ij}. The first subscript, *i*, denotes the node ID; whereas, the second subscript, *j*, denotes the

[1]SymTA/S is now acquired by Luxoft (https://www.luxoft.com).

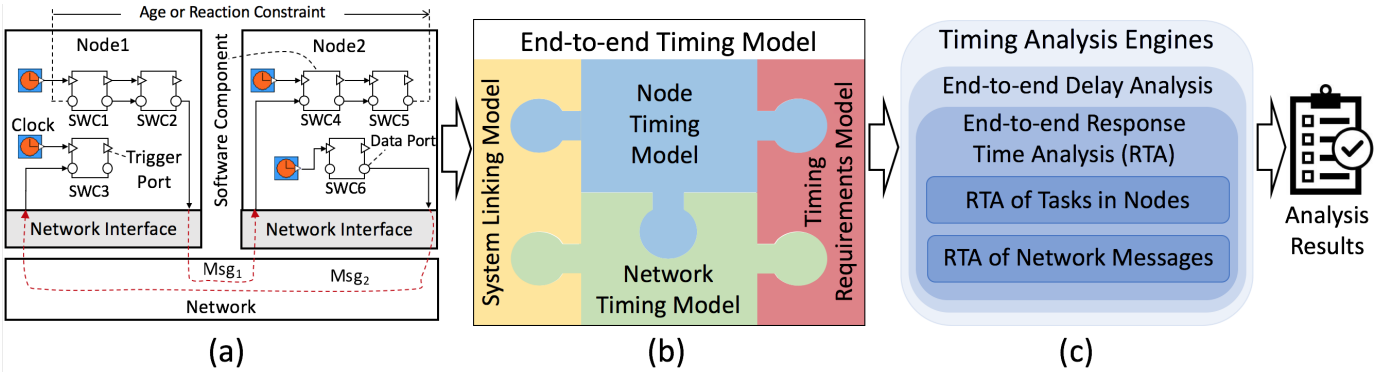

Fig. 1: (a) Example of a two-node distributed embedded system, (b) end-to-end timing model, and (c) timing analysis engines.

transaction ID. At the design time, a transaction corresponds to a chain of software components. A transaction can be time- or event-triggered with a period or a minimum inter-arrival time between two consecutive activations respectively, denoted by T_{ij}. A transaction contains $|\Gamma_{ij}|$ number of tasks. A task is denoted by τ_{ijk}, where the subscripts *i*, *j* and *k* represent the node, transaction and task IDs respectively.

$$\Gamma_{ij} := \langle \{\tau_{ijk}, \ldots, \tau_{ij|\Gamma_{ij}|}\}, T_{ij} \rangle \tag{1}$$

Each task can be activated periodically or sporadically with a defined period or minimum inter-arrival time respectively, denoted by T_{ijk}. Moreover, each task has a worst-case execution time (WCET) (C_{ijk}), an offset (O_{ijk}), a release jitter (J_{ijk}), a priority (P_{ijk}), a blocking time (B_{ijk}) and a worst-case response time (WCRT) (R_{ijk}) as given below.

$$\tau_{ijk} := \langle C_{ijk}, T_{ijk}, O_{ijk}, P_{ijk}, J_{ijk}, B_{ijk}, R_{ijk} \rangle \tag{2}$$

*2) **Network Timing Model**:* This model is based on a generic real-time network communication model that incorporates various in-vehicle networks including CAN [9], CANopen [45], HCAN [46], AUTOSAR COMM [47] and Ethernet. This model consists of one or more networks. Each network, $\mathcal{N}_i$, has a speed, denoted by f_i, and a set of messages, denoted by $\mathcal{M}_i$. In the case of point-to-point networks such as switched Ethernet, the network has one or more switches and various traffic classes. There are no switches and traffic classes in the case of broadcast networks like CAN. Any message belonging to $\mathcal{M}_i$ is represented by m_{ij}. The first subscript in m_{ij} denotes the ID of the message set or the ID of the corresponding network, whereas the second subscript provides the message ID. $\mathcal{M}_i$ is represented by the following tuple.

$$\mathcal{M}_i := \langle \{\mathcal{X}_{ij}, C_{ij}, \mathcal{F}_{ij}, P_{ij}, s_{ij}, T^P_{ij}, T^S_{ij}, \mathcal{L}_{ij}, J_{ij}, B_{ij}, R_{ij}\}, j = 1 \ldots |\mathcal{M}_i| \rangle \tag{3}$$

Where $\mathcal{X}_{ij}$, C_{ij}, $\mathcal{F}_{ij}$, P_{ij}, s_{ij}, T^P_{ij}, T^S_{ij}, $\mathcal{L}_{ij}$, J_{ij}, B_{ij} and R_{ij} represent the transmission type (periodic, sporadic or mixed), worst-case transmission time, frame type (standard or extended CAN frame), priority, data payload, period, minimum inter-arrival time, a set of links through which m_{ij} traverses in the network, jitter, blocking time and WCRT respectively.

*3) **System Linking Model**:* A distributed embedded system is often modeled with chains of tasks and messages, which may be distributed over two or more nodes. Fig. 1(a) shows examples of two chains that are distributed over two nodes. One chain is initiated by SWC1 in Node1 and terminated by SWC5 in Node2. The second chain is initiated and terminated by SWC6 and SWC3 respectively. In many component models such as RCM, data and control (trigger) flows are clearly separated as shown in Fig. 1(a). A SWC in the chain can be activated or triggered for execution by an independent source (e.g., SWC1 in Node1 is triggered by an independent clock) or by its predecessor SWC (e.g., SWC2 and SWC5). Any two neighbouring SWCs in a distributed chain can reside on the same node or on two different nodes. A message in the chain is triggered for transmission by the preceding SWC in the chain, also called the sending SWC (or sending task at runtime), in the case of passive networks like CAN or Ethernet AVB. However, a message can also be triggered by the network itself regardless of the sending task in the active networks like HaRTES [48]. All the control flow, data flow, mapping and linking information is included in the system linking model.

*4) **Timing Requirements Model**:* This model includes all the timing requirements in the system that includes (i) timing requirements on individual tasks, e.g., task deadlines (D_{ijk}), (ii) timing requirements on individual messages, e.g., message deadlines (D_{ij}), and (iii) timing constraints specified on the chains. The timing constraints conform to the timing model of the AUTOSAR standard [32], which defines eighteen timing constraints [39]. For example, two of these constraints, namely the age and reaction, are specified on a distributed chain in Fig. 1(a). Note that some timing requirements may have single constraining value, e.g., a task or message deadline has a single value. Whereas, the other timing requirements may have more than one constraining value, e.g., minimum and maximum values of the age constraint, in which case the corresponding age delay is required to be between the two values.

B. Support for TSN in the End-to-end Timing Model

The end-to-end timing model presented in the previous section is unable to incorporate TSN due to the new features of TSN as compared to the existing broadcast and point-to-point

onboard networks. Consequently, the network timing model, system linking model and timing requirements model need to be extended to support TSN.

*1) **Support for TSN Traffic Classes in the Network Timing Model**:* The network timing model needs to incorporate various types of new traffic classes that are supported by TSN. Let us denote the traffic class by $\mathcal{T}$. The domain of $\mathcal{T}$ is given as follows:

$$\mathcal{T} := \{ST, A, B, X, BE\} \tag{4}$$

Where ST represents the Scheduled Traffic. The two Stream Reservation (SR) classes are represented by A and B. Class A has a higher priority than Class B. X denotes the set of reserved classes, whose priorities are lower than the priority of class B but higher than the priority of the Best Effort (BE) class.

*2) **Support for TSN Switch and Link Models**:* The TSN switches are considered to be full-duplex, i.e., the input and output of a switch port are isolated. The connection between a node and a switch, as well as the connection between two switches is defined by a *link*. A link, denoted by l_{ij}, belongs to the network $\mathcal{N}_i$ and has a unique ID j. It is a directional connection, which means that there are two links associated to each physical port of the switch, one for the transmission and the other for the reception. The value of the network speed f_i remains the same in all links in the network. Let the set of all links and the total number of links in the network $\mathcal{N}_i$ be denoted by $\mathcal{L}_i$ and $|\mathcal{L}_i|$ respectively. The set $\mathcal{L}_i$ is represented as follows:

$$\mathcal{L}_i := \{l_{i1} \dots l_{i|\mathcal{L}|}\} \tag{5}$$

*3) **Support for Traffic Shaping in the TSN Link Model**:* The TSN standards consider various types of traffic shaping algorithms. In this work we focus on two of these algorithms (i) the credit-based shaping that applies only to the SR classes and (ii) the time-aware shaper that affects all traffic classes. For any SR traffic class A or B on a link l_{ij}, the credit replenishment rate (idleSlope) is denoted by $\mathfrak{R}^A_{ij}$ or $\mathfrak{R}^B_{ij}$ respectively Note that the messages belonging to the BE class do not undergo the credit-based traffic shaping. The time-aware shaping opens/closes the transmission gates of the different queues of a link following a table that specifies, at each point in time, which gates are open for transmission. The table is cyclically repeated.

*4) **Support for Preemption Mode in the TSN Link Model**:* The TSN standards allow to specify preemption mode for each traffic class on each individual link. Hence, each traffic class can be classified as the *express* or *preemptible* on each link. Let the preemption mode be denoted by $\mathcal{H}$. The domain of $\mathcal{H}$ is given as follows:

$$\mathcal{H} := \{Express, Preemptible\} \tag{6}$$

In order to support the preemption mode for each traffic class as well as the idleSlope in the SR traffic, the model of each link l_{ij} in Eq. 5 is represented as follows.

$$l_{ij} := \langle \mathcal{H}^{ST}_{ij}, \mathcal{H}^{A}_{ij}, \mathcal{H}^{B}_{ij}, \mathcal{H}^{X}_{ij}, \mathcal{H}^{BE}_{ij}, \mathfrak{R}^{A}_{ij}, \mathfrak{R}^{B}_{ij} \rangle \tag{7}$$

*5) **Extending the Message Model to Support TSN**:* The message set in Eq. 3 needs to be updated to support several properties that are specific to TSN. C_{ij} represents the worst-case transmission time of message m_{ij}, which depends on the message size and network speed. The header of each message is a constant value and is independent of the traffic class. A message may cross several links; the set of links that the message m_{ij} traverses is specified by $\mathcal{L}_{ij}$. The links in the set are ordered based on the order of message transmission, e.g., $\mathcal{L}_{ij} = \{l_{12}, l_{11}\}$ means that m_{ij} crosses first link l_{12} (link 2 in network 1) and then l_{11} (link 1 in network 1) between the sender and receiver nodes. Each message belongs to one of the traffic classes shown in Eq. 4. Hence, the model of a message m_{ij} in Eq. 3 should be updated with the $\mathcal{T}_{ij}$ parameter. Offsets are used to accommodate ST messages in the transmission schedule. The offset for each ST message is defined per link and the set of offsets for all links that m_{ij} traverses is specified in the set $\mathcal{O}_{ij}$, which contains offset-link pairs for the message. Let $|\mathcal{O}_{ij}|$ represent the size of this set. $\mathcal{O}_{ij}$ can be represented as follows:

$$\mathcal{O}_{ij} := \{(O_{ij1}, l_{i1}) \dots (O_{ij|\mathcal{L}_{ij}|}, l_{i|\mathcal{L}_{ij}|})\} \tag{8}$$

where O_{ij1} denotes the offset of m_{ij} on the first link (l_{i1}) along its path from its source to destination. Note that there is no offset defined for SR and BE traffic; hence, $\mathcal{O}_{ij}$ is an empty set for the messages that belong to these traffic classes.

*6) **Extending the Timing Requirements Model with Decomposed Message Deadlines**:* The message deadline, denoted by D_{ij}, defined in the timing requirements model corresponds to the constraint on the time interval from the queuing of the message in the network interface by the sending task to the delivery of the message at the network interface of the destination node. However, a message may traverse through several links in the case of TSN. Thus, the deadline decomposition per link for m_{ij} is required, which can be achieved either by dividing D_{ij} by the number of links that m_{ij} traverses or by distributing D_{ij} according to the load in each link. Let the per-link deadline of a message m_{ij} be denoted by d_{ij}. Then D_{ij} can be represented as follows:

$$\mathcal{D}_{ij} := \{(d_{ij1}, l_{i1}) \dots (d_{ij|\mathcal{L}_{ij}|}, l_{i|\mathcal{L}_{ij}|})\} \tag{9}$$

IV. Modeling Approach for TSN Communication

This section presents an approach to support modeling of TSN communication in distributed embedded systems. The approach comprises modeling of necessary hierarchical elements and TSN-specific network information, which any component model for distributed embedded systems should incorporate to support TSN. As a proof of concept, we extend an existing industrial component model, namely RCM, by implementing the proposed approach.

A. Required Elements in a Component Model to Support TSN

We identify several first-class structural elements, which are needed in any component model for vehicular distributed embedded systems to support modeling of TSN-based com-

munication and specification of corresponding timing information. Fig. 2 depicts these elements in a top-down hierarchical fashion. The top-level hierarchical element is the model of a distributed embedded system, e.g., X-by-wire system and anti-lock braking system. The system model consists of one or more models of each node and network. The node represents a unique run-time environment for one or more software applications. Each software application contains the software architecture that consists of SWCs and their interconnections. A SWC communicates with another SWC on the same or different node via its interface. Each SWC contains one or more behaviors, which are enriched with timing properties (e.g., WCET) and other resource requirements. Although there can be other elements within the structure hierarchy of the node in a component model such as cores, partitions, modes, assemblies and composites, the node-level elements shown in Fig. 2 are the basic elements required to model a node in a distributed embedded system.

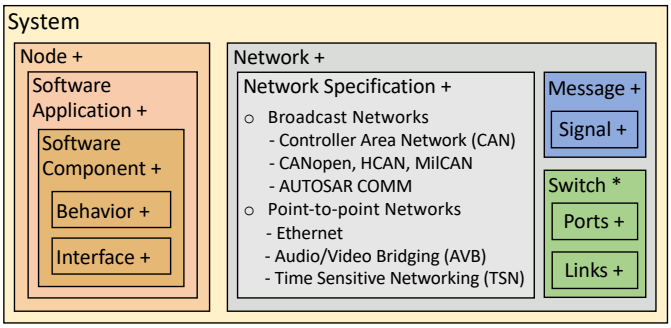

Fig. 2: Necessary structural hierarchy in a component model for distributed embedded systems to support modeling of TSN.

The network model consists of one or more Network Specification (NS) elements as shown in Fig. 2. The NS element is generic in the sense that it is able to incorporate many in-vehicle network protocols, including the broadcast networks like CAN, CANopen, HCAN, MilCAN, AUTOSAR COMM, and point-to-point networks like Switched Ethernet, AVB and TSN. The NS element includes the protocol-specific properties, configuration and corresponding timing information as discussed in Section III-A2. The network model also contains one or more messages. Each message further contains one or more signals. The signal-to-message mapping is specifiable in the NS element. In order to support the point-to-point networks, the network model contains one or more switches. The model of a switch is able to incorporate the properties of any switched Ethernet protocol including TSN. Each switch model is linked to a traffic shaper (discussed in the previous section), which is defined in the NS element. The switch model contains a set of ports and a set of links to support communication with the nodes and other switches.

B. Modeling of TSN Communication and a Proof of Concept

In this subsection we present an approach to explicitly model all the information discussed in Section III-B, including the timing information , which is required to model the TSN communication. In order to show a proof of concept for the proposed modeling approach, we extend the structural hierarchy of RCM by introducing/extending the missing elements and their properties, including the NS, message, switch and link elements. The extended network model in RCM to support TSN is shown in Figure 3(a). The properties of the network that can be defined by the user are shown in Figure 4(a). Note that each element modeled in RCM has a unique ID denoted by `Unique Identifier`. The user-specified properties of the network model include `Network Type`, `Name`, `Network ID`, and `Network Speed`. The `Network Type` property allows the user to select TSN or any other network protocol such as AVB and CAN. The `Name` and `Network ID` properties are unique. The model of a TSN switch in RCM and its user-specified properties are shown in Figure 3(b) and Figure 4(b) respectively. The user is able to select the type of switch such as TSN, AVB or general Ethernet switch using the `Switch Type` property. Moreover, the user can also specify the number of ports in the switch.

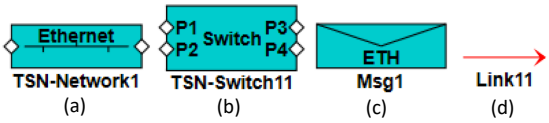

Fig. 3: RCM models of (a) TSN network, (b) TSN switch, (c) TSN message and (d) TSN link.

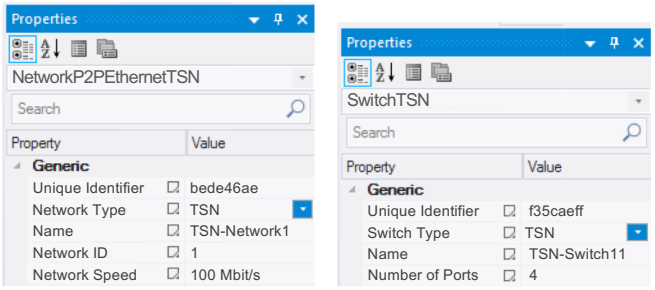

Fig. 4: Properties of TSN (a) network model, (b) switch model.

The extended model of the message in RCM and corresponding user-defined properties are shown in Figure 3(c) and Figure 5 respectively. The message model is independent of the type of Ethernet protocol. In fact, the properties of the message model are quite generic as they can support a message belonging to any broadcast or peer-to-peer onboard network protocol. For instance, the `Frame Type` property is specific to CAN and its higher-level protocols. This property is set to Not Applicable (N.A) in Figure 5, which represents the properties of a TSN message. Moreover, as the `Transmission Type` property is selected to be Periodic, the message can only have a period. Intuitively, the `Min Inter-arrival Time` property is set to N.A. In the

message model, the user can also specify an offset for each link which the ST class message traverses between its source and destination. Furthermore, there is an optional support for the user to specify per-link deadline of the message belonging to the ST class. If this information is not specified, the NS object automatically assigns the per-link deadlines according to the method discussed in Section III-B6.

Finally, the new link model in RCM and its user-defined properties are depicted in Figure 3(d) and Figure 6 respectively. The traditional onboard network protocols do not require any special properties for the links. However, in the case of TSN, the links not only provide connections between the nodes and switches or between switches, but also influence traffic shaping and message preemption modes as discussed in Section III-B. The traffic shaping in class A and B are supported by specifying the desired bandwidth (in Mbit/s) in the `Idle Slope Class A` and `Idle Slope Class B` properties respectively. Whereas, the preemption mode for each class in the link model is specified by selecting the desired mode (Express, Preemptable or Not set) in the `Preemption Mode` property of the corresponding class.

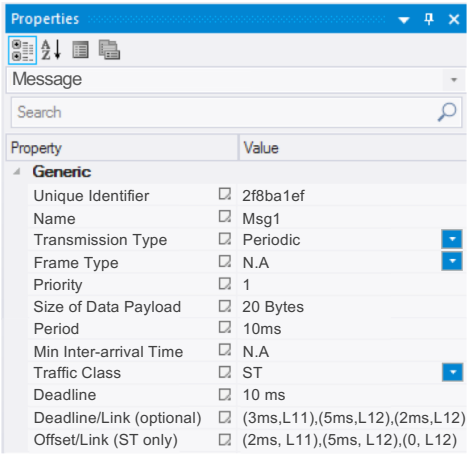

Fig. 5: Modeling of various properties of a TSN message.

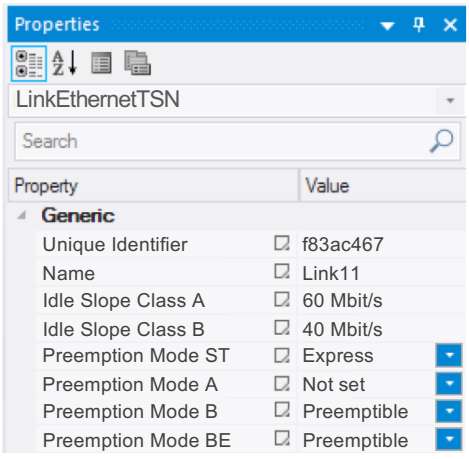

Fig. 6: Modeling of various properties of a TSN link.

V. Vehicular Application Use Case

This section validates the usability of the proposed modeling approach by modeling an industrial use case with extended RCM. Moreover, the sections conduct a partial end-to-end timing analysis of the modeled system.

A. Use Case Description

The use case considered in this paper is inspired by an industrial prototype system, which was originally developed by the BMW group [49] to show the possibility of using switched Ethernet as the backbone network in vehicles. The topology of the use case is depicted in Fig. 7. The topology consists of two networks, including a switched Ethernet network based on TSN standards and a legacy CAN network. The two networks are connected via a gateway ECU, which translates the CAN frames into Ethernet frames and vice versa. The TSN network is a multi-hop network as it consists of two interconnected TSN switches, one mounted in the front of the vehicle (Switch 1) and the other is mounted in the rear of the vehicle (Switch 2). A camera (CAM) is mounted on the vehicle that sends video frames and the GPS signals to the main processing ECU, known as the Head Unit. Moreover, a Proximity Sensor Handler (PSH) node is used to send proximity information to the Head Unit. There are three control nodes (Control 1-3), where one of them is connected to the TSN network, whereas the other two are connected to the CAN bus. The control nodes periodically send control messages with very small data payload to the Head Unit. The network speeds of TSN and CAN are set to 100 Mbit/s and 500 Kbit/s respectively.

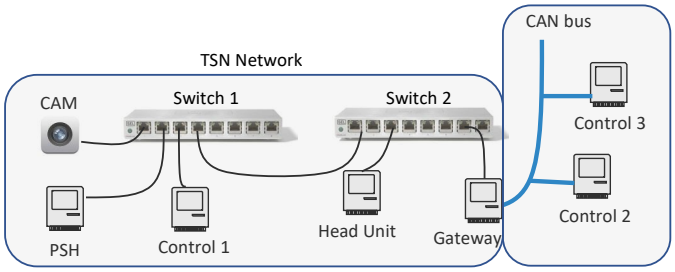

Fig. 7: Industrial use case: A vehicular distributed embedded system with heterogeneous networks.

Table I presents the traffic characteristics in the use case. The camera sends two different messages, i.e., the video frames and the GPS signals, thus the former is categorized as class A, while the latter is categorized as class B. The control and PSH messages are classified as ST class. Moreover, we consider the ST messages to be express, while the other traffic classes to be preemptible, e.g., the control and PSH messages can preempt the video (class A) and GPS (class B) messages. Furthermore, the idle slopes for classes A and B in all TSN links are set to 60 Mbit/s and 40 Mbit/s respectively. The CAN messages are assigned priorities (smaller the number, higher the priority) and once they enter the TSN network, they are converted to Ethernet messages and assigned the ST class by the gateway node.

Message name	Source	Destination	Size Bytes	Period ms	Class/ Priority
Msg1_Eth	CAM	Head Unit	3144	20	A
Msg2_Eth	CAM	Head Unit	500	50	B
Msg3_Eth	Control1	Head Unit	8	10	ST
Msg4_Eth	PSH	Head Unit	20	10	ST
Msg5_Eth	Gateway	Head Unit	8	10	ST
Msg6_Eth	Gateway	Head Unit	8	10	ST
Msg1_CAN	Control2	Gateway	8	10	1
Msg2_CAN	Control3	Gateway	8	10	2

TABLE I: Traffic characteristics in the case study.

B. Modeling of the Use Case with Extended RCM

The system-level software architecture of the use case is modeled with seven models of nodes, one model of TSN network, one model of CAN network, and interconnections among them using the extended RCM and Rubus-ICE tools suite, as depicted in Fig. 8. The internal model of the TSN network in RCM is shown in Fig. 9, which consists of two models of TSN switches and seven models of TSN links, identified as L11…L17. The properties of the links are specified according to the information presented in Section V-A. The internal software architectures of all nodes together with the modeled timing information in RCM are depicted in Fig. 10. The PSH node and each of the three control nodes, namely Control1, Control2 and Control3 include the model of only one SWC. The WCET of each of these SWCs is 500 μs. The SWCs in Control1 and PSH nodes send the ST messages Msg3_Eth and Msg4_Eth respectively. Whereas, the SWCs in Control2 and Control3 nodes send the CAN messages Msg1_CAN and Msg2_CAN respectively.

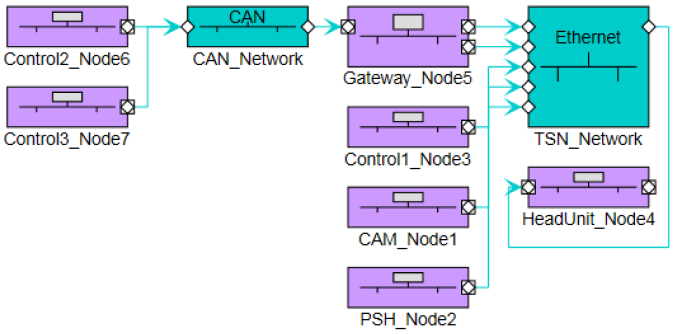

Fig. 8: System-level software architecture of the use case.

There are two SWCs in the CAM node, which send two Ethernet messages Msg1_Eth and Msg2_Eth. As SWC1 in this node performs heavy computations to process the video feed compared to SWC2 that processes GPS signals, SWC1 is assigned higher priority than SWC2. The WCETs of SWC1 and SWC2 in the CAM node are 2 ms and 500 μs respectively. The Gateway node also contains two SWCs, which receive the two CAN messages. In this node, the priority of SWC1 is higher than SWC2 because SWC1 receives and processes the higher priority message Msg1_CAN. The WCET of the two SWCs are 1 ms each. The Head Unit node also contains two SWCs. In this node, SWC1 receives and processes two Ethernet messages: Msg1_Eth and Msg2_Eth. Whereas, the SWC2 receives and processes four Ethernet messages: Msg3_Eth to Msg6_Eth. There are 12 timing constraints (6 of each Age and Reaction constraints) specified on 6 distributed chains in Fig. 10. Note that there are start and end objects associated to each timing constraint, defining the span of the constraint on the respective chain. The values of each Age and Reaction constraint are 20 ms and 30 ms respectively. The distributed chains and the values of their constraints are shown in Fig. 11.

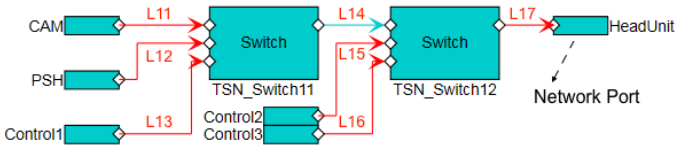

Fig. 9: Internal model of TSN network in the use case.

C. Partial End-to-end Timing Analysis of the Use Case

Although the focus of this paper is to provide a modeling technique for TSN and integrate it to the end-to-end modeling framework for distributed embedded systems, we take one step further by conducting a partial end-to-end timing analysis of the modeled use case. The analysis is performed using the timing analysis engines of Rubus-ICE tool suite, which implement the state-of-the-art timing analysis [41], [38]. The term "partial" means that the timing analysis can only be performed on the subsystem that includes the CAN network. Hence, only Chain1 and Chain2 shown in Fig. 11 are analyzed and the analysis results are shown in Table II. The calculated WCRT of each CAN message is equal to 1080 μs. It can be observed in Table II that the calculated Age and Reaction delays are smaller than the corresponding Age and Reaction constraints; hence, the timing constraints specified on Chain1 and Chain2 are satisfied.

Chain name	Age Constraint	Calculated Age Delay	Reaction Constraint	Calculated Reaction Delay
Chain1	20 ms	12580 μs	30 ms	22580 μs
Chain2	20 ms	13580 μs	30 ms	23580 μs

TABLE II: Timing analysis results of the partial use case.

The remaining four chains that are part of the TSN network cannot be analyzed for two reasons: (i) the existing response-time analysis for TSN is not yet mature despite several ongoing works in this regard [50], [51], and (ii) there is no existing technique to compute the end-to-end age and reaction delays in TSN-CAN heterogeneous networks.

VI. Conclusion

This paper has introduced a new modeling approach for TSN communication and integrated it within the model- and component-based software engineering framework for vehicular distributed embedded systems. The proposed approach

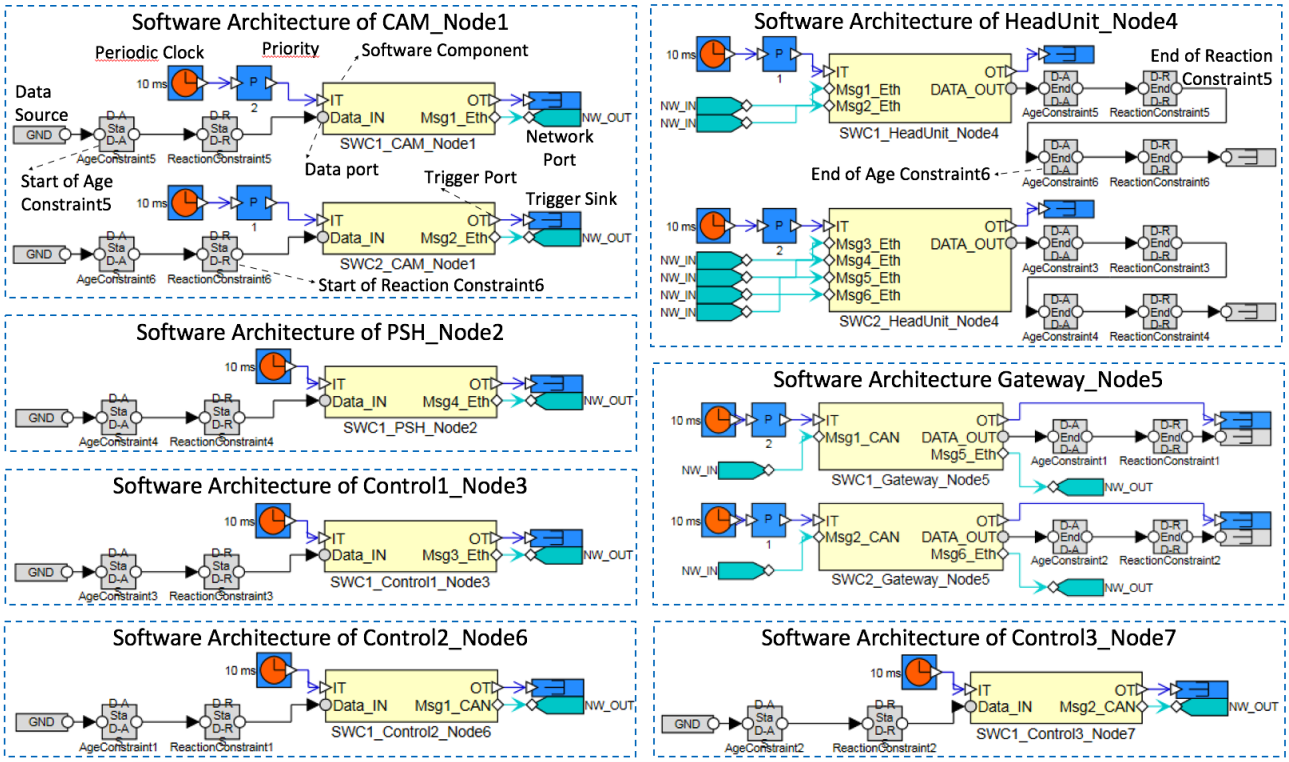

Fig. 10: Software architectures of the nodes along with the timing constraints specified on six distributed chains in RCM.

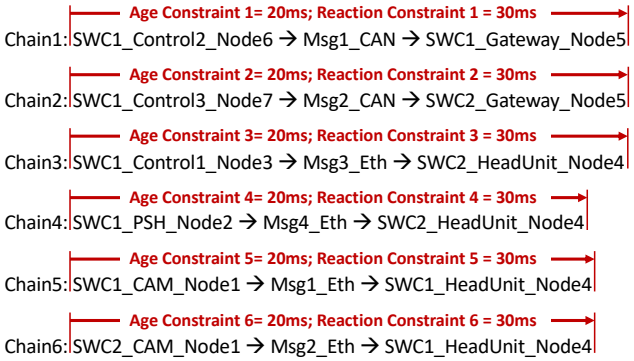

Fig. 11: Time-constrained distributed chains in the use case.

explicitly models the timing properties and requirements of TSN-based communication. The paper has also presented an end-to-end timing model that is augmented by the timing information of TSN. This model is required by the analysis engines to perform end-to-end timing analysis of the system. As a proof of concept, the proposed approach is implemented in an existing industrial component model, namely RCM, by introducing new first-class modeling elements and extending several existing elements. The new and existing modeling elements are backward compatible with the existing structure of the component model to support modeling of legacy communication protocols such as CAN. The proposed approach is generic enough to be integrated to any component model for distributed embedded systems that uses a pipe-and-filter style for communication between its software components.

In order to provide validation and show usability of the proposed approach, a use case from the vehicle industry is modeled with the extended RCM. The use case consists of a distributed embedded system with heterogeneous networks, including TSN and CAN. The paper also performs the end-to-end timing analysis of a sub-system of the use case that includes the nodes with CAN network. The subsystem that includes TSN cannot be analyzed because the existing response-time analysis for TSN is not yet mature and there is no existing end-to-end timing analysis framework for distributed embedded systems that contain TSN-CAN heterogeneous networks. The future work includes developing a method to extract timing models from the software architectures of distributed embedded systems modeled with the proposed approach.

Acknowledgement

The work in this paper is supported by the the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the DESTINE project and the Swedish Knowledge Foundation through the projects DPAC and HERO. We thank our industrial partners Arcticus Systems and Volvo Sweden.

References

[1] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356 –373, Feb 2007.

[2] Technavio, Global Automation Market in Automotive Industry–2018-2022, technical report SKU: IRTNTR20198, pp. 1-130, Jan, 2018, avialbale at: https://www.technavio.com.

[3] 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*, vol. 15, no. 2, 2019.

[4] G. Gut, C. Allmann, M. Schurius, and K. Schmidt, *Reduction of Electronic Control Units in Electric Vehicles Using Multicore Technology*, 2012, pp. 90–93.

[5] D. Reinhardt and M. Kucera, "Domain controlled architecture - a new approach for large scale software integrated automotive systems," in *International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2013)*, February 2013, pp. 221–226.

[6] M. Di Natale, "Design and development of component-based embedded systems for automotive applications," in *Reliable Software Technologies – Ada-Europe*. Springer Berlin Heidelberg, 2008, pp. 15–29.

[7] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. V. Chaudron, "A classification framework for software component models," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 593–615, 2011.

[8] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128 – 148, 2016.

[9] ISO Standard-11898, Road Vehicles–interchange of digital information–controller area network (CAN) for high-speed communication, 1993.

[10] P. Hank, S. Mller, O. Vermesan, and J. Van Den Keybus, "Automotive ethernet: In-vehicle networking and smart mobility," in *Design, Automation Test in Europe Conference Exhibition*, 2013, pp. 1735–1739.

[11] C. Varun and M. Kathiresh, "Automotive ethernet in on-board diagnosis (over ip) amp; in-vehicle networking," in *International Conference on Embedded Systems (ICES)*, July 2014, pp. 255–260.

[12] S. Brunner, J. Roder, M. Kucera, and T. Waas, "Automotive e/e-architecture enhancements by usage of ethernet tsn," in *13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2017, pp. 9–13.

[13] M. Ashjaei et al., "Modeling and timing analysis of vehicle functions distributed over switched ethernet", in *43rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Oct. 2017.

[14] IEEE Std. 802.1Q, IEEE Standard for local and metropolitan area networks, bridges and bridged networks, 2014.

[15] *IEEE Std. 802.1Qbv-2015: IEEE Standard for Local and metropolitan area networks, Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*, IEEE, 2015.

[16] *IEEE Std. 802.1Qbu-2016: IEEE Standard for Local and metropolitan area networks, Bridges and Bridged Networks - Amendment 26: Frame Preemption*, IEEE, 2016.

[17] IEEE, "IEEE standard for local and metropolitan area network–bridges and bridged networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, July 2018.

[18] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using ieee time-sensitive networks," *IET Cyber-Physical Systems: Theory Applications*, vol. 1, no. 1, pp. 86–94, 2016.

[19] M. H. Farzaneh and A. Knoll, "An ontology-based plug-and-play approach for in-vehicle time-sensitive networking (tsn)," in *7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Oct 2016, pp. 1–8.

[20] V. Gavrilut, Design Optimization of IEEE Time-Sensitive Networks (TSN) for Safety-Critical and Real-Time Applications, PhD Thesis: 2018-500, Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2018.

[21] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet tsn's time-aware and peristaltic shapers," in *IEEE Vehicular Networking Conference (VNC)*, Dec 2015, pp. 251–258.

[22] D. Thiele and R. Ernst, "Formal worst-case timing analysis of ethernet tsn's burst-limiting shaper," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 187–192.

[23] D. Maxim and Y.-Q. Song, "Delay analysis of avb traffic in time-sensitive networks (tsn)," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017, pp. 18–27.

[24] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of avb traffic in tsn networks using network calculus," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium*, 2018, pp. 25–36.

[25] Z. Li et al., "Time-triggered Switch-Memory-Switch Architecture for Time-Sensitive Networking Switches", in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[26] K. Hänninen et al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *IEEE Symposium on Industrial Embedded Systems*, 2008.

[27] S. Mubeen, H. Lawson, J. Lundbäck, M. Gålnander, and K. Lundbäck, "Provisioning of predictable embedded software in the vehicle industry: The rubus approach," in *4th IEEE/ACM International Workshop on Software Engineering Research and Industrial Practice*, May 2017.

[28] IEEE, "IEEE standard for local and metropolitan area network–bridges and bridged networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, July 2018.

[29] T. A. Henzinger and J. Sifakis, "The Embedded Systems Design Challenge," in *14th International Symposium on Formal Methods*, 2006.

[30] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.

[31] Peter Thorngren, keynote Talk: Experiences from EAST-ADL Use, EAST-ADL Open Workshop, Gothenberg, Oct., 2013.

[32] "AUTOSAR Techincal Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium, Oct., 2013," http://autosar.org.

[33] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A Component Model for Control-Intensive Distributed Embedded Systems," in *CBSE 2008)*, pp. 310–317.

[34] X. Ke, K. Sierszecki, and C. Angelov, "COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems," in *13th International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug. 2007.

[35] "Catalog of Specialized CORBA Specifications. OMG Group," http://www.omg.org/technology/documents/.

[36] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.

[37] M. H. Farzaneh, S. Shafaei, and A. Knoll, "Formally verifiable modeling of in-vehicle time-sensitive networks (tsn) based on logic programming," in *IEEE Vehicular Networking Conference (VNC)*, 2016, pp. 1–4.

[38] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Computer Science and Information Systems*, vol. 10, no. 1, 2013.

[39] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, and K.-L. Lundbäck, "Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints," *Software & Systems Modeling*, vol. 18, no. 1, pp. 39–69, Feb 2019.

[40] S. Kramer, D. Ziegenbein, and A. Hamann, "Real World Automotive Benchmarks for Free," in *6th Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

[41] 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 *CRTS Workshop*, dec. 2008.

[42] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *Journal of Systems Architecture*, vol. 80, pp. 104 – 113, 2017.

[43] S. Mubeen, M. Gålnander, J. Lundbäck, and K. Lundbäck, "Extracting timing models from component-based multi-criticality vehicular embedded systems," in *Information Technology - New Generations*, 2018.

[44] J. Palencia and M. G. Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets," *Real-Time Systems Symposium, IEEE International*, p. 26, 1998.

[45] "CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Version 4.02. February 13, 2002," http://www.can-cia.org/index.php?id=440.

[46] "Hägglunds Controller Area Network (HCAN), Network Implementation Specification," *BAE Systems Hägglunds, Sweden (internal document)*, April 2009.

[47] "AUTOSAR Requirements on Communication, Release 4.2.1," www.autosar.org, accessed on March 15, 2019.

[48] R. Santos, M. Behnam, T. Nolte, P. Pedreiras, and L. Almeida, "Multi-level hierarchical scheduling in ethernet switches," in *the 9th ACM International Conference on Embedded Software*, Oct 2011.

[49] H.-T. Lim, K. Weckemann, and D. Herrscher, *Performance Study of an In-Car Switched Ethernet Network without Prioritization*, 2011.

[50] M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, and L. LoBello, "Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support," *Real-Time Systems*, vol. 53, no. 4, 2017.

[51] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of avb traffic in tsn networks using network calculus," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2018.