End-to-end Timing Model Extraction from TSN-Aware Distributed Vehicle Software

Bahar Houtan*, Mehmet Onur Aybek†, Mohammad Ashjaei*, Masoud Daneshtalab*, Mikael Sjödin*, Saad Mubeen*
*Mälardalen University, Västerås, Sweden; †Arcticus Systems, Järfälla, Sweden
*firstname.surname@mdu.se; †onur.aybek@arcticus-systems.com

***Abstract*—Extraction of end-to-end timing information from software architectures of vehicular systems to support their timing analysis is a daunting challenge. To address this challenge, this paper presents a systematic method to extract this information from vehicular software architectures that can be distributed over several electronic control units connected by Time-Sensitive Networking (TSN) networks. As a proof of concept, the proposed extraction method is applied to an industrial component model, namely the Rubus Component Model (RCM), and its toolchain. Furthermore, the usability of the proposed method is demonstrated in an industrial use case from the vehicular domain.**

I. Introduction

The Time-Sensitive Networking (TSN) standards [1], [2], [3] have emerged as an attractive solution to address the high-bandwidth and real-time communication requirements in many Cyber Physiccal Systems (CPSs) in various domains such as vehicular and automation domains. These standards enhance switched Ethernet with deterministic traffic shaping mechanisms. More specifically, TSN supports hard real-time, high bandwidth with low-latency and low-jitter traffic transmission. These features are supported by offline scheduled time-triggered traffic enabled by the Time-Aware Shaper (TAS) mechanism, resource reservation for different classes of traffic by traffic shapers (CBS), and clock synchronization [4].

The software architectures of these CPSs, in particular in the vehicular domain, are described by various domain-specific languages or component models [5]. The modeling of timing information and its extraction from the software architectures is vital in supporting automated timing analysis of the systems. Even the modelling languages that allow functional modelling of distributed systems, with both software and communication models, typically use several different types of modelling concepts internally, e.g., different model concepts for software objects and for communication objects [6]. The use of different modelling languages and/or different modelling concepts makes the extraction of the end-to-end timing information difficult and time-consuming.

In this paper, we present an automated method to extract end-to-end timing information from the software architectures that are modelled with the Rubus Component Model (RCM) [7]. RCM is an industrial component model that supports modelling of various types of real-time networks, including TSN. We also integrate the proposed method to the model-based development tool-suite called Rubus-ICE [8] and demonstrate its applicability on a vehicular use case.

II. Background and Related Works

A. Modeling the Software Architecture (SWA)

There are several SWA modelling languages and component models [9], [10], [11], such as AUTOSAR, RCM, AMALTHEA, EAST-ADL, AADL, to mention a few. AUTOSAR is widely used for developing SWA for automotive systems. SymTA/s [12] is a commercial timing analysis and optimization framework that complies with AUTOSAR. A recent work in [13] integrates AUTOSAR adaptive with applicable standards to develop more sophisticated systems.

RCM comprises of hierarchical entities which are necessary to model a distributed embedded system that supports TSN. At the highest level, the system contains at least two nodes and a network element that interconnects the nodes. A node is a processing element that provides run-time environment for one or more Software Applications (SA). The SA provides spacial and temporal isolation to the part of overall SWA within the system. A SWA is modelled by interconnecting a set of Software Components (SWCs). An SWC is a design-time entity that corresponds to a task at run-time or in the timing model. The SWCs communicate with each other by their interfaces (a set of data and trigger ports).

To the best of our knowledge, RCM is the first and only component model that supports comprehensive modeling of TSN [10]. Recently, there have been some efforts in increasing the performance and applicability of the other modeling approaches to RCM by model transformation [14]. The work in [15] proposes a mapping technique between AMALTHEA and RCM to enable timing analysis of AMALTHEA-based models in RCM. In addition, the work in [16] presents a mapping from EAST-ADL models to RCM with the aim of enabling the timing analysis of a non-RCM model.

B. Timing models and timing analysis

Mubeen et al. [17] proposed modeling and extraction of timing models from SWAs of component-based multi-criticality embedded systems while considering several onboard communication protocols like Controller Area Network (CAN), CANopen, HCAN, AUTOSAR COMM and generic switched Ethernet. However, it does not TSN, which is the main focus of our work. The work in [10] complements [17] by describing several aspects of modeling TSN standards from timing analysis perspective. However, the presented timing model and extraction method is not automated. In this paper,

we address this gap in automated extraction of end-to-end timing information for TSN.

Vehicular software architectures are often modelled with chains of SWCs (tasks at runtime) and network messages. In order to verify the timing requirements on these chains, end-to-end data-propagation delays (age and reaction) should be calculated and compared with the corresponding age and reaction constraints [18], [19]. The age delay is the time elapsed between the arrival of data at the input of the chain and the latest availability of the corresponding data at the output. Whereas, the reaction delay corresponds to the earliest availability of the data at the output of the chain corresponding to the data that just missed the read access (of the event) at the input of the chain.

There are several works that present the end-to-end timing analysis [20], [21]. The work in [22], [23] considered the end-to-end timing analysis for distributed embedded systems that are based on CAN network. The work in [24] considers the systems that use Ethernet networks. The works in [20] and [25] presented the end-to-end timing analysis for automotive applications, where some of the techniques have been implemented in tools to support component-based software development, e.g., [22] and [26]. The works in [27], [8] present open research challenges and their solutions when integrating the timing analysis with model-based development tools. However, these works only focus on traditional onboard networks such as CAN without considering TSN. In the case of TSN, there are several additional features that need to be considered such as synchronization of the nodes, and the presence of different shaping mechanisms for different TSN classes. A recent work in [28] shows that the existing end-to-end timing analysis supports the non-scheduled TSN classes, but it does not support the scheduled traffic in TSN.

III. End-to-end Timing Model Extraction Method

The timing model extraction method bridges a TSN-aware end-to-end timing model with the model- and component-based software development framework for distributed embedded systems. An end-user is able to interact with the component model directly by means of modelling the SWA. The end-user is typically a software modeller/developer, with limited or no knowledge about detailed timing information in the system. In order to retrieve the required timing information to analyse and verify the modelled SWA, we propose a timing model extraction method that is conceptualized in Fig. 1.

The timing model extraction module in Fig. 1 coordinates a set of methods to extract the end-to-end timing model. On one hand, some information can be directly obtained from the SWA. On the other hand, some information needs to be extracted through the configuration step, where the retrieved information from the component model is processed either by the system integrator/configurator or by the automated configuration tools. Furthermore, at the integration layer all the retrieved information from the previous steps are prepared to be extracted into the end-to-end timing model according to the following classification.

1) **User-defined (User):** The properties in this category are extracted from the input specified by the end-user while developing the SWA. The end user is the software developer or software modeler with an abstract overview of the system. The end-user does not have any information about the timing aspects of the system.
2) **Software-architecture-derived (SWA-d):** The properties that are either inherited from other components in the software-architecture; or are calculated according to the user-defined properties; or implicitly initialized based on other properties.
3) **Configurable (Conf.):** This category holds the parameters obtained from the SWA, which are configured according to some logical constraints, and algorithms for the sake of optimizing timing performance of the system. Such parameters could also be defined by system experts based on their knowledge of the system requirements, i.e., system configurators or integrators.
4) **Analysis-derived (Analysis):** The values of the parameters in this category are obtained by performing various analyses, e.g., response-time analysis of individual nodes, response-time analysis of TSN network, and end-to-end timing analysis response-time analysis.

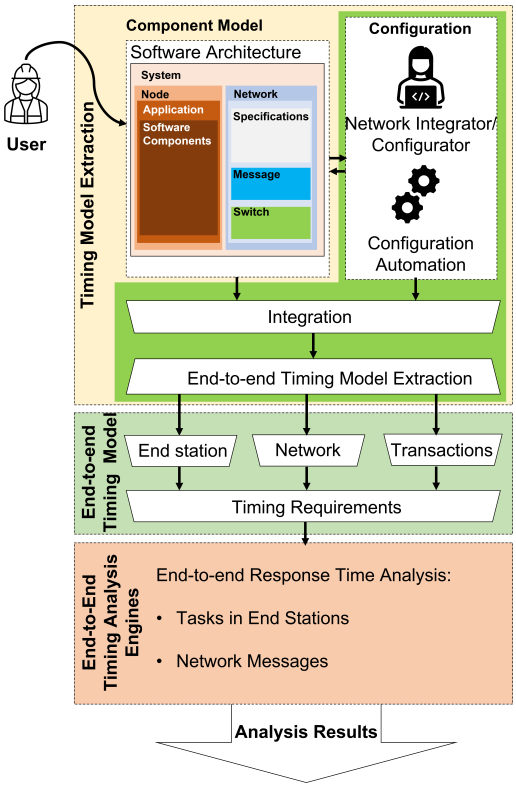

Fig. 1: Model Extraction Method.

The mapping of the parameters in the system timing model to each category is presented in Table I.

IV. Evaluation with an Automotive Use-case

The end-to-end timing model extraction method presented in the previous section is implemented as a proof of concept in the Rubus-ICE tool suite. In this section, we take advantage

TABLE I: Extracting timing model for TSN.

Component	Parameters	User	SWA-d	Conf.	Analysis
Node	Application	✓			
	Application Criticality		✓		
SWC	WCET	✓			
	Period	✓	✓		
	Offset		✓	✓	
	Priority	✓			
	Jitter				✓
	Blocking				✓
	Response Time				✓
	Deadline		✓	✓	
Network	Speed		✓		
	Link Set		✓		
	Class Set		✓		
	slope_A		✓	✓	
	slope_B		✓	✓	
	Preemption		✓	✓	
Message	Transmission Mode		✓		
	Transmission Time		✓		
	Priority		✓		
	Payload	✓			
	Period		✓		
	Path Links		✓	✓	
	Jitter				✓
	Blocking				✓
	Response Time				✓
	Offset per Link		✓	✓	
	Deadline		✓	✓	
SWC Chain	Transaction end/start	✓			
	Data path	✓			
	Period		✓		
	Age Delay				✓
	Reaction Delay				✓
	Timing Constraints	✓		✓	

of a real automotive use-case to validate the applicability of the proposed method. For the sake of evaluations, we first model a SWA of a distributed embedded system, consisting of a set of transactions, in RCM. Then we extract the end-to-end timing model from the SWA in Rubus-ICE. Using the extracted model, we perform the end-to-end timing analysis of the SWA using the end-to-end timing analysis implemented in Rubus-ICE [28] and discuss the analysis results.

A. Use-case Setup

In the use-case setup, we assume the Camera node is capable of generating TSN traffic, whereas HU and AVSink nodes only receive TSN traffic from the Camera node. There are 2 distributed chains (transactions). Each transaction includes tasks from two different nodes and one message between the nodes. The initiator node has only one task that sends the message to the network. The transaction terminator node includes two tasks. On the receiver's side, the first task receives the TSN message and activates the second task in the receiver node. We assume that all tasks in an node belong to the same application. The WCET of all the task within the system are set to 0,05 ms. The offsets of the sender tasks within the system are set to the default value which is 0. The period of all the tasks in the transaction 1 are set to 10 ms. Task 1 of transaction 1 has the highest priority within the Camera node, with the priority value of 2. The message inherits the period of the Task 1 of transaction 1 and the payload size of the message is 1542 $Bytes$. The message is transmitted utilizing the TSN class ST. The offset of the message is configured as 0.013 ms.

Furthermore, the period of all the tasks within the transaction 2 are set to 20 ms. Task 1 of transaction 2 has is the lowest priority task in the Camera node (priority 1) and it is transmitting a message with the payload size 1542 $Bytes$ utilizing class BE. The BE message has no offset.

Finally, the idle slopes are set to 0 for all the links since the AVB classes are not used in the use-case. The reaction and age constraints on all the transactions are subsequently 70 ms and 45 ms. These constraints are specified by expert integrators from the industry (providers of the use-case).

B. Modelled use-case in Rubus-ICE

The system-level SWA of the use-case modeled with RCM is depicted in Fig. 2. The system-level SWA consists of 3 node models. All the nodes are connected to one TSN network model, as shown in Fig. 2. In the internal model of the TSN network as shown in Fig. 3, the flow from all the sender nodes are either towards HU or AVSink (the only sink nodes within the system). The SWA of each node is depicted in Fig. 4, where the RCM representation of the set of SWCs within each node is shown by yellow components. For example, the SWA of the Camera node consists of two SWC, where SWC1 in the Camera node that is used in transaction 1 has period of 10 ms and sends an ST message with the priority of 2. Besides, SWC2 in the Camera node is used in transaction 2 and has a lower priority than SWC1 (priority 1). The period of SWC2 is 20 ms. Transaction 1 is initiated from the SWC1 in the Camera node and is terminated at the SWC2 in the HU node, which has the priority of 4. The terminator task of the transaction is triggered by the SWC1 in the HU node. As a result, the terminator task inherits the period of its predecessor task, namely SWC1 in the HU node.

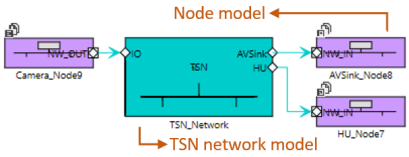

Fig. 2: System-level SWA of the use-case.

C. End-to-end Response-Time Analysis Results

The results of the end-to-end timing analysis include reaction and age delays of the transactions as well as response-times of the network messages. For instance, transaction 1 which contains an ST message with the response-time of 0,025 ms. The age and reaction delays of transaction 1 are subsequently 20,100 and 30,100 ms. Transaction 2 is also initiated at the Camera node, though from a different SWC, namely (SWC2), and it is terminated at SWC4 in HU. Transaction 2 uses the class BE of the TSN network. The calculated response-time of the message in Transaction 2 is 0,153 ms. Besides, the age and reaction delays of transaction 2 are subsequently 40,300 and 60,300 ms. As the specified age and reaction constraints on all transactions are 70 ms and