Integrating Response-time Analysis for Heterogeneous Networks with Rubus Analysis Framework: Challenges and Preliminary Solutions

Saad Mubeen, Mohammad Ashjaei and Thomas Nolte
MRTC, Mälardalen University, Västerås, Sweden
{saad.mubeen, mohammad.ashjaei, thomas.nolte}@mdh.se

John Lundbäck and Kurt-Lennart Lundbäck
Arcticus Systems AB, Järfälla, Sweden
{john.lundback, kurt.lundback}@arcticus-systems.com

***Abstract*—In this paper we discuss the challenges that are faced when the state-of-the-art research results are transferred to a model-based tool chain for the industrial use. These challenges are often overlooked when the research results are implemented in an academic environment. In particular, we discuss various challenges regarding the implementation and integration of the response-time analysis for heterogeneous networks, comprising of CAN and Ethernet AVB, as a plug-in for the Rubus Analysis Framework. Rubus tool suite is used for the model- and component-based development of software for vehicular real-time systems by several international companies. We also discuss preliminary solutions to deal with the challenges.**

I. Introduction

Nowadays, a high-end car contains more than five different types of in-vehicle networks. Some of these networks are used to connect Electronic Control Units (ECUs) in an in-vehicle system that requires hard real-time network communication (i.e., a deadline violation can result in the system failure). For example, Controller Area Network (CAN) [1] is used to connect ECUs in an adaptive cruise control system. On the other hand, there are other networks that are used in those in-vehicle systems that require soft real-time communication among the ECUs (i.e., an occasional deadline miss may be tolerated by the system). For example, Ethernet Audio Video Bridging (AVB) [2] with a high-throughput support of up to 100 Mbit/s, has found its application in the vehicle domain, e.g., in the infotainment system. In this paper, our main focus is on the heterogeneous networks, where CAN and Ethernet AVB are interconnected via a gateway.

The developers of real-time systems are required to provide evidence for the system's predictable behavior. The research community has developed a plethora of a priori schedulability analysis techniques to provide such an evidence. One of the most powerful, well-established and industrially accepted schedulability analysis techniques is the Response-Time Analysis (RTA) [3], [4]. It is a method to calculate upper bounds on the response times of execution entities such as tasks and messages in a real-time system and network respectively.

A. Problem Statement

When a new timing analysis technique such as RTA is developed in an academic environment, a lot of issues concerning its implementation and integration with a model-based tool chain are overlooked. For instance, the input parameters for the analysis are either hard coded or assumed to be available as inputs (e.g., in a text file). There is no focus on how the analysis interacts (extracts input information) with the modeling environment that is used to develop the software architecture of the system or with other models and tools in a model-based tool chain. At best the analysis is often implemented as a standalone tool that gets inputs either directly from the user or from text files. In fact, there is not much to gain by focusing on all these issues if the analysis is meant to be used in the academic environment.

On the other hand, if the newly developed analysis is to be transferred to the industry then several challenges related to its implementation and integration with an industrial model-based tool chain must be solved. Within this context, in addition to the above mentioned issues, other challenges include, e.g., unambiguous extraction of timing information; extensions in the existing modeling approach (used by the modeling tools) to support a seamless integration of the new analysis with the tool chain; handle the implementation requirements that are dictated by the legacy tools that are included in the tool chain; and various levels of testing. All these challenges must be solved before the analysis can be used in the industry.

B. Motivation and Paper Contribution

In this paper we discuss the integration of RTA for heterogeneous networks including Ethernet AVB and CAN-AVB gateway as a Rubus-ICE [5] plug-in. The Rubus-ICE tool suite is used for model- and component-based development of real-time systems in the vehicle industry. We discuss various challenges that we face during the implementation of the RTA in the tool suite. We also discuss preliminary solutions to deal with these challenges. There are several commercial tools that support RTA for Ethernet AVB, e.g., SymTA/S [6]. However, to the best of our knowledge, none of these tools openly reveal the implementation and integration related problems, solutions and experiences. Whereas, this paper explicitly highlights such challenges and their preliminary solutions. We believe, the problems and corresponding solutions that we discuss in this paper are equally applicable when any other real-time analysis is transferred to any industrial model-based tool chain.

II. Background and Related Work

A. Rubus Concept and its Analysis Framework

The Rubus suite, developed by Arcticus Systems[1], provides a collection of methods and tools for model- and component-based development of vehicular real-time systems. It has been in industrial use for over 20 years. Today, it is used by several international companies, e.g., Volvo Construction Equipment, BAE Systems, Knorr-bremse, Mecel and Hoerbiger.

978-1-4673-7929-8/15/$31.00 © 2015 European Union

[1]http://www.arcticus-systems.com

The Rubus concept is based around the Rubus Component Model [7] and its development environment, called Rubus-ICE, that includes modeling tools, code generators, analysis tools and run-time infrastructure. Fig. 1 shows various tools in the Rubus-ICE tool suite. A real-time application is modeled in the Designer. The application is compiled by the compiler to the Intermediate Compiled Component Model (ICCM). The builder integrates a bunch of plug-ins that operate on the ICCM model. The purpose of the Rubus plug-in framework is to allow the implementation of any method or technique in isolation and support its integration as an add-on plug-in with the integrated development environment. The requirements on a plug-in include the specification of supported system model, required inputs, provided outputs, error handling support and a user interface as shown in Fig. 2. Some of the plug-ins constitute the Rubus Analysis Framework. The timing analysis supported by the Rubus Analysis Framework includes RTA for CAN and its higher-level protocols; RTA for tasks in a single node; and end-to-end response-time and delay analyses in distributed real-time systems [8]. Finally, the analyzed and verified model of the application is used to automatically generate code by the coder tool. It should be noted that there is no support in Rubus for analyzing Ethernet AVB and multiple networks that are connected by gateways.

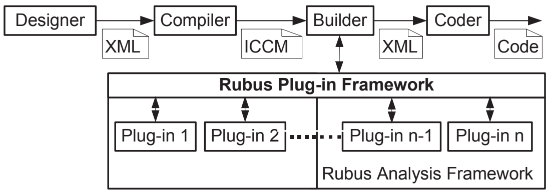

Fig. 1. Several tools in Rubus-ICE supporting design to code generation.

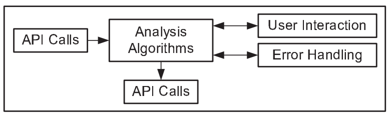

Fig. 2. Conceptual organization of the Rubus-ICE plug-in.

B. *Related Analyses and Tools*

Ethernet AVB relies on a Credit-based Shaping Algorithm (CBSA), which regulates the traffic transmission. It categorizes the traffic into two classes, class A and B, where class A has higher priority than class B. If two or more messages have the same priorities, the arbitration is carried out using the FIFO policy. There are several timing analysis approaches that have been proposed for the Ethernet AVB network, e.g., [9]. However, these analyses are restricted to the computation of worst-case response time per class, without distinguishing the messages response time. The RTA for messages in the Ethernet AVB architecture is given in [10]. However, the work in [11] shows that the RTA in [10] considers only one blocking factor that results from lower priority messages, which is not the case in Ethernet AVB due to the traffic shaper. Thus, it proposes a new RTA that takes the new blocking term into account. However, the analysis is limited to the constrained deadline traffic model and a single-switch architecture. In this work we implement the RTA for Ethernet AVB, developed in [11], as a plug-in in Rubus-ICE.

In [8], [12], the authors discuss the issues and experiences concerning the implementation of end-to-end timing analysis in an industrial tool. However, the focus in these works is on the support for CAN; multiple networks are not supported. Whereas, in this paper we focus on the issues and challenges that are faced when RTA for heterogeneous networks is implemented and integrated with a model-based tool chain.

SymTA/S is a tool for model-based timing analysis and optimization. It supports RTA of various vehicular networks including CAN, Flexray, AFDX, and Ethernet AVB. py-CPA [13] is a compositional performance analysis tool that supports the end-to-end timing analysis. MPS-CAN Analyzer[2] is an academic tool that supports RTA for CAN and Ethernet AVB. To the best of our knowledge, none of these tools share the problems, solutions and experiences of implementing and integrating the analysis which they support today.

III. Implementation Challenges, Experiences and Preliminary Solutions

This section discusses the challenges that we faced during the implementation and integration of the RTA as a plug-in in Rubus-ICE. We also discuss the preliminary guidelines and solutions to deal with these challenges.

A. *Stringent Requirements Dictated by the Legacy Tools*

The purpose of implementing the RTA for Ethernet AVB is to not only support the analysis of Ethernet messages but also allow the user to analyze distributed chains (composed of tasks and messages) that span over several ECUs and multiple networks including CAN and Ethernet AVB as shown in Fig. 3. When end-to-end response times are calculated in distributed real-time systems, the response times of network messages depend upon the response times of tasks in the ECUs and vice versa. The end-to-end response times are calculated using the attribute (jitter) inheritance algorithm[14]. This means that the difference between the worst- and best-case response times of the sending task is inherited as a jitter by the message. Similar attribute inheritance is performed on messages and their receiving tasks. This algorithm is iteratively repeated until converging response times are obtained in all the ECUs and networks or the deadlines (if specified) are violated.

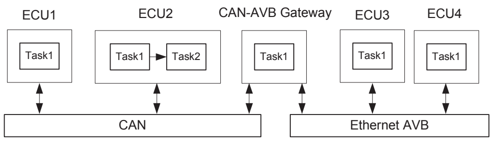

Fig. 3. Multiple networks in a distributed real-time system.

Often there are stringent requirements, for the implementation and integration of new analysis, which are dictated by the model-based tool chain. For instance, the Rubus Analysis Framework allows the integration of new analysis as a plug-in which is interfaced with the builder tool as shown in Fig. 1. The plug-ins can only be executed in a sequence, i.e., the next plug-in can execute only when the previous plug-in has finished its execution and has provided the analysis results back to the builder tool. The plug-in framework in Rubus-ICE

[2]https://github.com/saadmubeen/MPS-CAN

is very general and is similar to plug-in framework of any other industrial tool. In order to calculate the end-to-end response times, RTA of Ethernet AVB depends upon the previously implemented analyses in Rubus-ICE that support RTA for CAN and ECUs. Hence, the new plug-in must interact with the existing plug-ins. We need to find out the most suitable implementation approach that is able to minimize the running time of the analysis framework.

From the implementation point of view, the most suitable approach is to implement the new analysis as a standalone plug-in; integrate it with the builder tool; and run it sequentially with the other plug-ins as shown in Fig. 4(a). However, there is another implementation overhead. That is, another plug-in must be created that implements the attribute inheritance algorithm and glues the new and the related existing plug-ins together. This plug-in is identified as the Holistic Plug-in in Fig. 4(a). Another downside of this approach is that several plug-ins need to run in a sequence iteratively until converging response times of the messages are obtained. As a consequence, the analysis framework may take several minutes before it can provide the analysis results of a medium-sized real-time application. It can take even longer for larger applications. Such delays in the running time of the analysis engines are highly undesirable in the industrial environment.

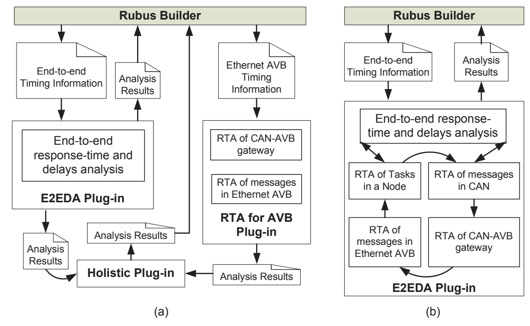

Fig. 4. Options to integrate the RTA for Ethernet AVB with Rubus-ICE.

In order to significantly reduce the running time of the analysis engines, we select a more complicated approach for the implementation of the RTA as shown in Fig. 4(b). Using this approach, the RTA for Ethernet AVB is implemented within the existing end-to-end response-time and delays analysis (E2EDA) plug-in. This option does not support any reuse of the existing standalone plug-ins. It requires relatively large implementation time because the implementer has to understand the analysis, implementation and code of the existing plug-in. Moreover, this option requires higher amount of time-to-test because the implementation of the existing plug-in must also be verified after the integration of the new RTA.

B. Extensions Required in the Modeling Approach for a Seamless Integration of the RTA

For a model-based tool chain to support the RTA for Ethernet AVB, it should also support the modeling of the network. Rubus-ICE already supports the modeling of real-time networks by means of a modeling object called Network Specification (NS). There are two parts of the NS: (1) protocol-independent and (2) protocol-specific. The protocol-independent part is independent of the properties and communication rules of the network protocol. It specifies network properties (e.g, network speed); message properties; data elements that are mapped to the messages; and a list of sender and receiver ECUs. The protocol-specific part of the NS is uniquely specified for each protocol. It defines the behavior semantics of each message according to the protocol; captures frame properties; and specifies the information regarding signal length; signal age; and mapping, encoding, packing and decoding of signals from/to messages. In conclusion, we provide the above information to Rubus-ICE for developing a modeling profile that corresponds to the protocol-specific part of the NS for the Ethernet AVB protocol.

C. Modeling and Timing Analysis of CAN-AVB Gateway

Various CAN-Ethernet gateways have been proposed in the existing literature [15]. These gateways use different strategies to forward the messages. In order to use the network resources efficiently, the gateways collect several CAN frames and map them to a single Ethernet packet. Recently, a CAN-AVB gateway is proposed [16], that follows the characteristics of the Ethernet AVB, such as periodic transmission and the CBSA. The proposed CAN-AVB gateway uses different scheduling policies to collect the incoming CAN frames, including FIFO, fixed-priority and Earliest Deadline First (EDF) policies. The information regarding the gateway strategies is required for the RTA. In order to get this information, we build a modeling profile for the gateway in the protocol-specific part of the NS. The selection of the most suitable gateway strategy for CAN-AVB gateway needs further investigation.

D. Unambiguous Extraction of Timing Information

This challenge is, often, a non-issue if the RTA is implemented for academic use. This is because all the input information required by the RTA is either hard coded or assumed to be available as an input file. However, if the RTA is to be implemented as a part of an industrial model-based tool chain, all the inputs required by the RTA, specifically the timing information, must be unambiguously extracted from the software architecture of the real-time application that is modeled in a separate tool within the tool chain. Hence, the implementer of the analysis plug-in has to not only implement the RTA, but also support the extraction of unambiguous timing information automatically from the modeling tool. It is possible that the modeling tool may contain redundant timing information. Often, the design model contains redundant timing information, e.g., the message periods can be defined by the user or inherited from the sender tasks in the case of a standalone network or a distributed system respectively. Such information duplication may lead to inconsistency in the models. We implement special routines as part of the newly implemented plug-in to check such redundant information. Similarly, other timing attributes that are checked for redundancy include worst- and best-case execution times, minimum-update times (for sporadic transmission), offsets, priorities (priority and ID of a message may or may not be the same depending upon the network protocol) and jitter. In order to extract the linking and mapping (between messages and signals and vice versa) information within the distributed chains, we use the method presented in [8].

E. Verification and Validation of the Implemented RTA

A significant part of the newly implemented plug-in consists of error handling and sanity checking routines. These routines detect and isolate faults and present them to the user during the analysis. For example, these routines evaluate the validity of all inputs, intermediate results that are iteratively inherited as inputs, variable overflow and overload conditions. In order to verify and validate the newly implemented analysis, our test plan includes several levels of testing. At the first level we perform the standalone testing, i.e., testing the RTA implementation (code) in isolation. At the second level, we integrate the RTA implementation with the MPS-CAN Analyzer (which is an academic tool) and perform testing to identify potential errors and bugs. Finally, we perform the integration testing which refers to the testing of the RTA implementation after it has been integrated with the Rubus Analysis Framework. The integration testing is a heavy weight and a time consuming activity. The reason is that we have to not only test the RTA for AVB but also test the implementation of the end-to-end response time and delay analysis due to our choice of implementation approach (see Section III-A). At all these levels of testing, the input test cases are hard coded; read from external files; acquired from test generation engines; manually inserted in the ICCM file (see Fig. 1); and automatically extracted from the software architecture of the modeled application.

F. Collaboration between the Integrator and Implementer

In our experience, we have observed that there is a close and continued collaboration required between the integrator and implementer of the plug-in. This is because a plug-in is developed in isolation by the implementer (with research background). Whereas, it is integrated with the model-based tool chain by an integrator from the industry with a limited knowledge about the schedulability analysis. Consequently, the two roles need to continuously communicate with each other throughout the integration and verification process.

G. Feedback to Improve the Schedulability of the System

If the modeled system is unschedulable, as indicated by the analysis results, the newly implemented plug-in can provide suggestions to guide the user to make the system schedulable. However, it is very difficult to provide such feedback because there can be several reasons behind the system being not schedulable. In particular, for the heterogeneous traffic, that traverses from CAN to AVB, the gateway configurations affect the timing analysis. For instance, the gateway may use a fixed-size buffer to collect the CAN frames. In addition, Ethernet frames can also be generated upon the expiry of a timer. These two parameters affect the delay of the messages in the gateway. We plan to implement an algorithm, based on the RTA, to find the most suitable values for setting the gateway parameters in order to decrease the response times of the messages. It should be noted that the CAN-AVB gateway is normally implemented within an ECU, hence it is possible to adjust the parameters like the buffer size and timer using the algorithm.

IV. Summary of Ongoing Work and Conclusion

We have performed the standalone testing of the new plug-in. The integration of the plug-in with the tool suite is ongoing. Currently we are investigating various gateway strategies that can be used in the CAN-AVB gateway. Our goal is to model and implement the most suitable gateway strategy in terms of lower gateway delays for the global traffic that traverses through heterogeneous networks.

In this paper we have discussed various challenges that are faced when state-of-the-art research results such as RTA for the Ethernet AVB network and CAN-AVB gateway are implemented and integrated with an industrial model-based tool chain. We also discussed preliminary solutions and guidelines to deal with these challenges. As a proof of concept, we integrate the RTA as a plug-in for the analysis framework of Rubus-ICE tool suite. Rubus-ICE is used for the model- and component-based development of vehicular real-time systems by several international companies. We believe, our findings and experiences can be useful in guiding the implementer during the implementation of other complex real-time analysis techniques in any industrial model-based tool chain that supports a plug-in framework for the integration of new tools.

Acknowledgement

The work in this paper is supported by the Swedish Foundation for Strategic Research (SSF) and ARTEMIS within the projects PRESS and CRYSTAL. The authors thank the industrial partners Arcticus Systems AB and Volvo Sweden.

References

[1] Robert Bosch GmbH, “CAN Specification Version 2.0,” postfach 30 02 40, D-70442 Stuttgart, 1991.

[2] “Audio/video bridging task group of ieee 802.1, available at http://www.ieee802.org/1/pages/avbridges.html.”

[3] M. Joseph and P. Pandya, “Finding response times in a real-time system,” *Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[4] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, “Fixed priority pre-emptive scheduling: an historic perspective,” *Real-Time Systems*, vol. 8, no. 2/3, pp. 173–198, 1995.

[5] “Rubus ICE-Integrated Development Environment,” http://www.arcticus-systems.com.

[6] A. Hamann, R. Henia, R. Racu, M. Jersak, K. Richter, and R. Ernst, “Symta/s - symbolic timing analysis for systems,” 2004.

[7] K. Hänninen et.al., “The Rubus Component Model for Resource Constrained Real-Time Systems,” in *3rd IEEE International Symposium on Industrial Embedded Systems*, June 2008.

[8] 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.

[9] J. Imtiaz, J. Jasperneite, and L. Han, “A performance study of ethernet audio video bridging (avb) for industrial real-time communication,” in *IEEE Conference on Emerging Technologies Factory Automation*, September 2009.

[10] J. Diemer, D. Thiele, and R. Ernst, “Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching,” in *7th IEEE International Symposium on Industrial Embedded Systems*, June 2012.

[11] U. Bordoloi, A. Aminifar, P. Eles, and Z. Peng, “Schedulability analysis of ethernet avb switches,” in *20th International Conference on Embedded and Real-Time Computing Systems and Applications*, August 2014.

[12] S. Mubeen, J. Mäki-Turja, and M. Sjödin, “Implementation of Holistic Response-Time Analysis in Rubus-ICE: Preliminary Findings, Issues and Experiences,” in *The 32nd IEEE Real-Time Systems Symposium (RTSS), WIP Session*, December 2011, pp. 9–12.

[13] pyCPA Tool, http://pycpa.readthedocs.org/, accessed Mar. 2015.

[14] K. Tindell and J. Clark, “Holistic schedulability analysis for distributed hard real-time systems,” *Microprocess. Microprogram.*, vol. 40, pp. 117–134, April 1994.

[15] A. Kern et al, “Gateway strategies for embedding of automotive CAN-Frames into Ethernet-Packets and vice versa.” in *Architecture of Computing Systems*. Springer, 2011, pp. 259–270.

[16] C. Herber, A. Richter, T. Wild, and A. Herkersdorf, “Real-time capable can to avb ethernet gateway using frame aggregation and scheduling,” in *Design, Automation Test in Europe Conference Exhibition*, March 2015.