Implementation of End-to-End Latency Analysis for Component-Based Multi-Rate Real-Time Systems in Rubus-ICE

Saad Mubeen*, Jukka Mäki-Turja*† and Mikael Sjödin*
*Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden
†Arcticus Systems, Järfälla, Sweden
{saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se

Abstract

One of the most important activities during the development of multi-rate real-time systems is to analyze the end-to-end timing. In this paper, we discuss the implementation plan and preliminary work regarding the integration of end-to-end latency analysis for component-based multi-rate real-time systems (both in single-node and distributed systems) within the analysis framework of Rubus-ICE. The Rubus-ICE is an existing industrial tool suite for component-based development of distributed real-time embedded systems. Further, we present the implementation methodology to integrate two end-to-end latency semantics, i.e., data age and data reaction within the existing holistic response-time analysis plug-in of Rubus-ICE.

1 Introduction

Often, an embedded system needs to interact and communicate with its environment in a timely manner, i.e., the embedded system is a real-time system. For such a system, the desired and correct output is one which is logically correct as well as delivered within a specified time. Many real-time systems are also safety critical which means that the system failure can result in catastrophic consequences such as endangering human life or the environment. In safety-critical real-time systems, a logically correct but late response is as bad as logically incorrect response.

The multi-rate systems are those systems in which the tasks are activated with different periods. Multi-rate real-time applications can be realized in single-node as well as in distributed real-time systems. These systems often have chains of tasks. The tasks in these chains are activated independent of each other with different periods. These systems find their applications in many domains such as automotive, aerospace, robotics, industrial control, etc. In this work we will focus only on the automotive or vehicular domain. In [6], the authors have a view that almost all automotive embedded systems are multi-rate systems.

1.1 Motivation

In order to provide evidence that each action in the system will meet its deadline, *a priori* analysis techniques such as schedulability analysis have been developed by the research community. Response Time Analysis (RTA) [3, 16] is a schedulability analysis technique to calculate upper bounds on response times of tasks or messages in a real-time system. In a multi-rate real-time system, merely computing Worst Case Response Times (WCRTs) and comparing them with corresponding deadlines is not sufficient to predict the complete timing behavior of the system. Due to over- and under-sampling in these systems, some values in the buffers may be over-written by new values and the effect of the old values may never propagate at the output. Moreover, it is also possible to have several duplicates of the output. Hence, it is also important to compute age of the data and first reaction to the inputs. The end-to-end latency refers to the time elapsed between the arrival of a signal at the first task and production of actuation signal (in response to the input signal) by the last task in the chain [15]. In single-rate real-time systems, tasks in a chain are not activated by independent events, in fact, there is only one activating event. Hence, end-to-end response times and end-to-end latencies will have equal values. On the other hand, these values are not the same for multi-rate real-time systems. Therefore, a complete analysis of multi-rate real-time systems requires the computation of not only WCRTs but also end-to-end latencies.

1.2 Goals and Paper Contribution

We discuss our plan and preliminary work for the implementation of end-to-end latency analysis [6] of component-based multi-rate real-time applications in both single-node and distributed systems. Our goal is to implement and integrate this analysis within the analysis framework of an existing industrial tool suite, Rubus-ICE [1] that is used for component-based development of Distributed Real-time Embedded (DRE) systems. We discuss an implementation methodology to integrate two end-to-end latency semantics, i.e., Data Age and Data Reaction [6, 17] within the existing Holistic Response Time Analysis (HRTA) plug-in [12] in Rubus-ICE. The methodology supports ease of implementation, integration and evaluation by allowing the reuse of HRTA that is already implemented and pre-tested in Rubus-ICE.

2 Background and Related Work

2.1 The Rubus Concept

Rubus is a collection of methods and tools for model- and component-based development of dependable embedded real-time systems. Rubus is developed by Arcticus

978-1-4673-0692-8/12/$31.00 ©2012 IEEE

Systems [1] in close collaboration with several academic and industrial partners. Rubus is today mainly used for development of control functionality in vehicles. The Rubus concept is based around the Rubus Component Model (RCM) [8] and its development environment Rubus-ICE (Integrated Component development Environment), which includes modeling tools, code generators, analysis tools and run-time infrastructure. The overall goal of Rubus is to be aggressively resource efficient and to provide means for developing predictable and analyzable control functions in resource-constrained embedded systems. RCM expresses the infrastructure for software functions, i.e., the interaction between software functions in terms of data and control flow separately. The control flow is expressed by triggering objects such as internal periodic clocks, interrupts, internal and external events. In RCM, the basic component is called Software Circuit (SWC). The execution semantics of an SWC is simply: upon triggering, read data on data in-ports; execute the function; write data on data out-ports; and activate the output trigger. Recently, RCM is extended to support the development of DRE systems [13].

The Rubus Analysis Framework (RAF). The Rubus model allows expressing real-time requirements and properties at the architectural level. For example, it is possible to declare real-time requirements from a generated event and an arbitrary output trigger along the trigger chain. For this purpose, the designer has to express real-time properties of SWCs, such as Worst Case Execution Times (WCETs) and stack usage. The scheduler will take these real-time constraints into consideration when producing a schedule. For event-triggered tasks, response-time calculations are performed and compared to the requirements. RAF supports distributed holistic response-time analysis and shared stack analysis.

2.2 Response Time Analysis (RTA)

RTA of Tasks in a Node. RTA is used to perform a schedulability test which means it checks whether or not tasks in the system will satisfy their deadlines. Tindell [20] developed the schedulability analysis for tasks with offsets for fixed-priority systems. It was extended by Palencia and Gonzalez Harbour [14]. Mäki-Turja and Nolin [9] reduced pessimism from the offset-based RTA. In [6, 17], the authors point out that the existing analysis does not target general multi-rate systems and register-based communication.

RTA of Messages in a Network. In this paper, we will focus only on Controller Area Network (CAN) [7] and its high-level protocols for real-time network communication. Tindell et al. [19] developed the schedulability analysis of CAN. It was revisited and revised by Davis et al. [4]. In [5], Davis et al. extended the analysis for CAN network with a mix of priority- and FIFO-queued nodes. In [10], Mubeen et al. extended the existing analysis to support RTA of mixed messages in CAN.

Holistic RTA (HRTA). It combines the analysis of nodes and a network. Hence, it computes the response times of event chains that are distributed over several nodes in a DRE system. We consider the HRTA that corresponds to the holistic schedulability analysis for DRE systems [18].

2.3 End-to-End Latency Analysis

Stappert et al. [17] formally described end-to-end timing constrains for multi-rate systems in automotive domain. In [6], Feiertag et al. presented a framework for the computation of end-to-end latencies for multi-rate automotive embedded systems. Furthermore, they emphasized on the importance of two end-to-end latency semantics, i.e., "maximum age of data" and "first reaction" in control systems and body electronics respectively. A scalable technique, based on model checking, for the computation of end-to-end latencies is described in [15]. In this work, we will implement the end-to-end latency semantics [6] in RAF.

3 End-to-End Latencies in Component-Based Multi-rate Systems

In this section, we discuss and graphically illustrate the end-to-end latencies in single-node and distributed multi-rate real-time systems that are developed using model- and component-based development approach.

3.1 Single-node Multi-rate Real-time Systems

A single-node multi-rate real-time system modeled with RCM is shown in Figure 1. There are three SWCs, i.e., SWC_A, SWC_B and SWC_C in the node. These SWCs are activated by independent clocks with different periods, i.e., 8ms, 16ms and 4ms respectively. SWC_A reads the input signals from the sensors while SWC_C produces the output signals for the actuators. Assume that each SWC will be allocated to an individual task by the run-time environment generator. Also assume that WCET of each task is one time unit.

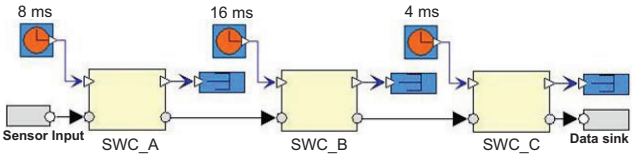

Figure 1. A multi-rate real-time system in a single node modeled in RCM

The time line corresponding to the run-time execution of the three tasks (corresponding to three SWCs) is depicted in Figure 2. It can be seen in the figure that there are multiple outputs corresponding to a single input signal. The four end-to-end latency semantics are identified in Figure 2.

Last In First Out (LIFO). This latency is equal to the time elapsed between the current non-overwritten release of task τ_A (i.e., input) and corresponding first response of task τ_C (i.e., output).

Last In Last Out (LILO). This latency is equal to the time elapsed between the current non-overwritten release of task τ_A and corresponding last response of task τ_C. This latency is identified as "Data Age" in [6]. Data age specifies the longest time data is allowed to age from production by the initiator until the data is delivered to the terminator. This latency finds its importance in control applications where the interest lies in the freshness of the produced data.

First In First Out (FIFO). This latency is equal to the

time elapsed between the previous non-overwritten release of task τ_A and first response of task τ_C corresponding to the current non-overwritten release of task τ_A. This latency is identified as "Data Reaction" in [6]. Data reaction specifies the longest allowed reaction time for data produced by the initiator to be delivered to the terminator. This latency finds its importance in the body electronics domain where first reaction to an input is important.

First In Last Out (FILO). This latency is equal to the time elapsed between the previous non-overwritten release of task τ_A and last response of task τ_C corresponding to the current non-overwritten release of task τ_A.

We are specifically interested in the implementation of two of these latencies, i.e., data age and data reaction because of their importance in the automotive and body electronic control systems. One of the main goals of this work is to implement the algorithms [6] for the computation of these two latency semantics in RAF.

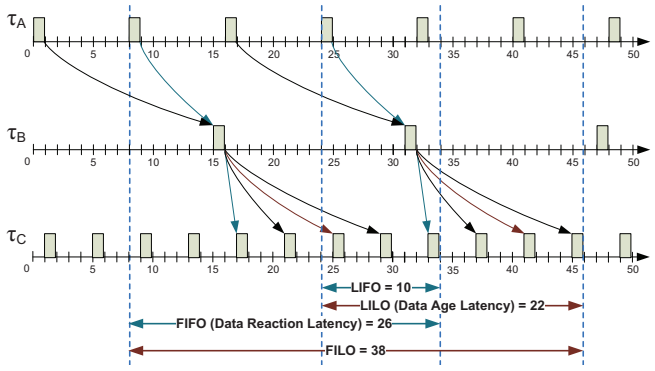

Figure 2. End-to-end latencies in multi-rate systems

3.2 Multi-rate Distributed Real-time Systems

Consider a model of a two-node multi-rate DRE system modeled with RCM as shown in Figure 3. The nodes are connected to a CAN network. The internal model of the nodes is also shown in Figure 3.

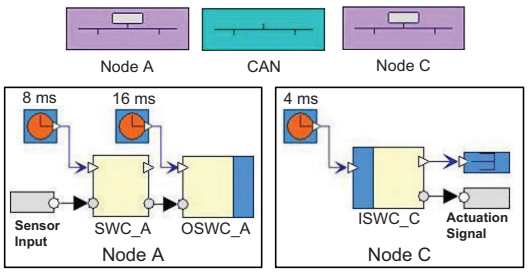

Figure 3. A multi-rate distributed real-time system modeled in RCM

In Node A, SWC_A is triggered by a clock with a period of 8ms. The $OSWC\_A$ component that is responsible for sending a message to the network is triggered by another clock with a period of 16ms. The $ISWC\_C$ is a component that receives a message from the network and is activated by a clock with a period of 4ms. Assume that each component is allocated to a separate task at run-time, i.e., the components SWC_A, $OSWC\_A$ and $ISWC\_C$ are allocated to tasks τ_A, τ_B and τ_C respectively. Since, the system consists of tasks with similar activation patterns and periods as compared to the tasks in single-node multi-rate real-time system example discussed in the previous subsection, it can be scheduled in a similar manner as indicated by τ_A, τ_B and τ_C in Figure 2. Moreover, the end-to-end latencies in multi-rate distributed real-time systems are defined in a similar fashion.

4 Implementation Methodology

In this section, we discuss the implementation methodology to integrate the end-to-end latency analysis for component-based multi-rate real-time systems [6] (developed in TIMMO project [2]) within the HRTA plug-in available in the Rubus-ICE tool suite.

The analysis in [6] implicitly requires the computation of response times of individual tasks, messages and event chains. For example, the calculation of four end-to-end latencies for the multi-rate real-time system shown in Figure 1 requires the calculation of the response time of the task τ_C (corresponding to the component SWC_C). Similarly, the calculation of four end-to-end latencies for the multi-rate DRE system shown in Figure 3 requires the calculation of the response time of the task τ_C in node C. However, the response time of τ_C is dependent upon the response time of the message received from the CAN bus.

Since, the existing HRTA plug-in in Rubus-ICE is able to calculate response times of tasks, network messages and event chains, we propose to reuse the previously implemented and tested analysis for the calculations of end-to-end latencies as shown in Figure 4. If a single-node multi-rate real-time system is under analysis then the end-to-end latency algorithms use services from Node RTA implementation alone by means of API calls as shown by a dotted-line bi-directional arrow in the extended HRTA plug-in in Figure 4. Similarly, if a multi-rate DRE system is under analysis then the end-to-end latency algorithms use services from the Node, Network and Holistic algorithms as shown by the dotted-line bi-directional arrows in the extended HRTA plug-in.

The holistic response times of distributed transactions are computed using HRTA algorithm [18]. It iteratively runs the algorithms for node and network analysis. The response times of tasks in a node are computed using tighter RTA of tasks with offsets [9]. The response times of CAN messages are computed using a general analysis that supports several high-level protocols for CAN. It also includes the analysis of CAN that we recently extended for mixed messages [10, 11]. The network analysis consists of four profiles:

1. RTA of CAN [19, 4].
2. RTA of CAN for mixed messages [10].
3. RTA of CAN for FIFO queues [5].
4. RTA of mixed CAN messages with priority and FIFO queues [11].

The first two profiles are already implemented and are part of the HRTA plug-in. Whereas, the last two profiles

will be implemented in the extended HRTA Plug-in. To the best of our knowledge, Rubus-ICE will be the first tool suite to implement the tighter version of offset-based RTA [9], RTA of CAN for mixed messages [10] and RTA of mixed CAN messages with priority and FIFO queues [11] as part of the end-to-end latency analysis.

The extended HRTA Plug-in will be interfaced with the Rubus builder tool. The end-to-end timing and tracing information will be extracted from the Intermediate Compiled Component Model (ICCM) shown in Figure 4. The working principle of HRTA algorithm is as follows. In the first step, release jitter of all messages and tasks in the system is assumed to be zero. The response times of all messages in the network and all tasks in each node are computed. In the second step, attribute inheritance is carried out. This means that each message inherits a release jitter equal to the difference between the worst-case and best-case response times of its sender task (computed in the first step). Similarly, each receiver of a message inherits a release jitter equal to the difference between the worst-case and best-case response times of the message (computed in the first step). In the third step, response times of all messages and tasks are computed again. The computed response times are compared with the previous response times (from the first step). The analysis terminates if the current and previous response times are equal or a deadline (if specified) is exceeded, otherwise, these steps are repeated.

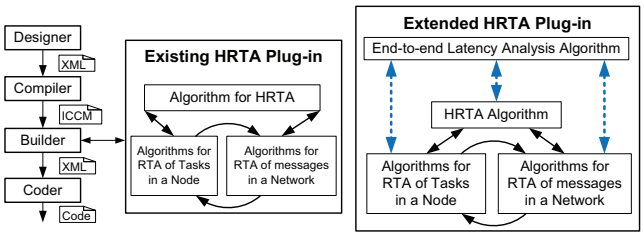

Figure 4. Integration of end-to-end latency analysis in HRTA plug-in

5 Summary

We discussed the work in progress on the implementation of end-to-end latency analysis for component-based multi-rate real-time systems (both in single-node and distributed systems) in the analysis framework of an existing industrial tool suite. We discussed an implementation methodology to integrate two end-to-end latency semantics, i.e., data age and data reaction within the existing holistic response-time analysis plug-in of Rubus-ICE. The methodology provides ease of implementation by supporting the reuse of implemented and pre-tested existing analysis. Currently, we are implementing the end-to-end latency analysis in Rubus-ICE according to the proposed methodology. Once the integration testing is over, we will conduct an industrial case study to provide a proof-of-concept implementation. We plan to discuss the implementation issues and experiences in the extended version of this paper. We also plan to implement the analysis of other network protocols (e.g., Flexray, switched ethernet, etc.) and integrate them within the extended HRTA plug-in.

Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) within the project FEMMVA. The authors would like to thank the industrial partners Arcticus Systems and Volvo Construction Equipment (VCE).

References

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

[2] TIMMO Methodology , Version 2. *TIMMO (TIMing MOdel), Deliverable 7*, October 2009. The TIMMO Consortium.

[3] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority pre-emptive scheduling:an historic perspective. *Real-Time Systems*, 8(2/3):173–198, 1995.

[4] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35:239–272, 2007.

[5] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka. Controller Area Network (CAN) Schedulability Analysis with FIFO queues. In *23rd Euromicro Conference on Real-Time Systems*, July 2011.

[6] 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 *Compositional Theory and Technology for Real-Time Embedded Systems, 2008. CRTS 2008. Workshop on*, dec. 2008.

[7] R. B. GmbH. CAN Specification Version 2.0. Postfach 30 02 40, D-70442 Stuttgart, 1991.

[8] 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.

[9] J. Mäki-Turja, , and M. Nolin. Tighter response-times for tasks with offsets. In *Real-time and Embedded Computing Systems and Applications Conference (RTCSA)*. Springer-Verlag, August 2004.

[10] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Extending schedulability analysis of controller area network (CAN) for mixed (periodic/sporadic) messages. In *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA),*, sept. 2011.

[11] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Response-Time Analysis of Mixed Messages in Controller Area Network with Priority- and FIFO-Queued Nodes. In *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, May 2012.

[12] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Support for Holistic Response-time Analysis in an Industrial Tool Suite: Implementation Issues, Experiences and a Case Study. In *19th IEEE Conference on Engineering of Computer Based Systems (ECBS)*, pages 210 –221, April 2012.

[13] S. Mubeen, J. Mäki-Turja, M. Sjödin, and J. Carlson. Analyzable Modeling of Legacy Communication in Component-Based Distributed Embedded Systems. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2011*, pages 229 –238, Sep. 2011.

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

[15] A. C. Rajeev, S. Mohalik, M. G. Dixit, D. B. Chokshi, and S. Ramesh. Schedulability and end-to-end latency in distributed ecu networks: formal modeling and precise estimation. In *Proceedings of the tenth ACM international conference on Embedded software*, EMSOFT '10, pages 129–138. ACM, 2010.

[16] L. Sha, T. Abdelzaher, K.-E. A. rzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28(2/3):101–155, 2004.

[17] F. Stappert, J. Jonsson, J. Mottok, and R. Johansson. A Design Framework for End-To-End Timing Constrained Automotive Applications. In *Embedded Real-Time Software and Systems (ERTS), 2010*.

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

[19] K. Tindell, H. Hansson, and A. Wellings. Analysing real-time communications: controller area network (CAN). In *Real-Time Systems Symposium (RTSS) 1994*, pages 259 –263.

[20] K. W. Tindell. Using offset information to analyse static priority preemptively scheduled task sets. Technical Report YCS 182, Dept. of Computer Science, University of York, 1992.