Translating End-to-End Timing Requirements to Timing Analysis Model in Component-Based Distributed Real-Time Systems

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*—Often, component-based real-time systems are modeled with trigger and data chains. The end-to-end timing requirements on trigger chains are different from those on data chains. For a trigger chain, the interest lies in the calculation of holistic response time and its comparison with end-to-end deadline. Whereas, the schedulability of a data chain requires a comparison between its end-to-end latencies and corresponding deadlines. We discuss the problem of translating end-to-end timing requirements unambiguously from component-based real-time systems into timing analysis models which are required as input by the analysis tools. We also provide preliminary guidelines for such translations in the existing industrial tool suite.**

***Keywords*-Response-time analysis; end-to-end timing analysis; timing model; component-based development.**

I. Introduction

Often, component-based real-time systems are modeled with chains of components that are translated to chains of tasks at run-time. A task chain is a sequence of more than one tasks in which every task (other than the first) receives a trigger, data or both from its predecessor. One way to classify these chains is as trigger chains and data chains. In trigger chains, there is only one triggering source (e.g, event, clock or interrupt) that activates the first task in the chain which, in turn, triggers the next task and so on. On the other hand, data chains have independent source of triggering for every task. Each task (except first) in these chains receives data from its predecessor. In component-based real-time systems, the timing requirements such as end-to-end deadlines on trigger and data chains are specified in the component model.

The safety-critical nature of many real-time systems requires evidence that the actions by the system will be provided in a timely manner, i.e., each action will be taken at a time that is appropriate to the environment of the system. Therefore, it is important to make accurate predictions of the timing behavior of such systems. For this purpose, a priori analysis techniques such as schedulability analysis have been developed by the research community. The analysis tools operate on the timing analysis model which should be extracted from the modeled application. The end-to-end timing requirements should be unambiguously translated to the analysis model from the component model of the real-time application.

In this paper, we discuss the problem of translating the end-to-end timing requirements into analysis model from single-node as well as distributed real-time systems that are developed using component-based approach. We also provide preliminary guidelines for such translations in the industrial tool suite, Rubus-ICE, used for component-based development of distributed real-time systems.

II. Background and Related Work

A. Response Time Analysis (RTA)

RTA [1], [2] is a powerful, mature and well established schedulability analysis technique. It is a method to calculate upper bounds on response times of tasks (or messages) in a real-time system (or a network). RTA applies to systems where tasks are scheduled with respect to their priorities and which is the predominant scheduling technique used in real-time operating systems today [3].

1) RTA of tasks with offsets: Tindell [4] developed the response-time analysis for tasks with offsets for fixed-priority systems. It was extended by Palencia and Harbour [5]. Mäki-Turja and Nolin [6] reduced pessimism from the offset-based RTA. In this work we will consider the tighter version of the offset-based RTA [6] as part of the end-to-end response-time and latency analysis.

2) RTA of Messages in a Network: In this paper, we will focus only on Controller Area Network (CAN) [7] and its high-level protocols. Tindell et al. [8] developed the schedulability analysis for CAN. It was revisited and revised by Davis et al. [9]. In [10], Davis et al. extended the analysis for CAN network with a mix of priority- and FIFO-queued nodes. In [11], [12], Mubeen et al. extended the existing analysis to support RTA of mixed messages in CAN with priority- and FIFO-queued nodes. Later on, Mubeen et al. [13] extended the existing analysis for CAN to support mixed messages that are scheduled with offsets. In this work we will consider all of the above analysis as part of the end-to-end response-time and latency analysis.

3) Holistic RTA (HRTA): It calculates the upper bounds on the response times of event chains that may be distributed over several nodes in a distributed real-time system. It combines the analysis of tasks in nodes and messages in a network. We consider the HRTA that corresponds to the analysis in [14].

B. End-to-End Latency Analysis

Stappert et al. [15] formally described end-to-end timing constrains in automotive domain. In [16], Feiertag et al. presented a framework for the computation of end-to-end latencies for multi-rate automotive embedded systems. They emphasized on the importance of two end-to-end latencies, i.e., "maximum age of data" and "first reaction" in control systems and body electronics domains respectively. A scalable technique for the computation of end-to-end latencies based on model checking is described in [17]. In this work, we will consider the analysis discussed in [16].

C. 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 Systems [18] 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) [19] 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 distributed real-time systems [20], [21].

1) 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.

III. Research Problem

A. Problem Statement

A component-based real-time system can be modeled with trigger chains (see Figure 4), data chains (see Figure 1) or a combination of both. The end-to-end timing requirements on trigger chains are different from those on data chains. If the system is modeled with trigger chains then the interest, from the schedulability point of view, lies in the calculation of their end-to-end or holistic response times. Hence, the end-to-end deadline requirements placed on trigger chains correspond to holistic response times. In order to check the schedulability of such systems, the holistic response times are compared to the corresponding deadlines. If the holistic response times of all trigger chains are less than or equal to the corresponding deadlines, the system is considered schedulable.

On the other hand, merely computing the holistic response times and comparing them with corresponding end-to-end deadlines is not sufficient to predict the complete timing behavior of the real-time system that is modeled with data chains. There may be over and under sampling in a real-time system due to data chains with independent and varying clock periods for individual tasks. This may cause some values in the data buffers to be over written by new values and hence, the effect of the old values may never propagate to the output. Further, it is also possible to have several duplicates of the output. In such systems, the end-to-end timing requirements, especially in automotive domain [16], are placed on the first reaction to input and age of the data at output. Hence, it is also important to compute the end-to-end latencies (or delays) in such systems. 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 a data chain [17].

In a real-time system that contains only trigger chains, tasks in a chain are not activated by independent events, in fact, there is only one activating event in the chain. Hence, holistic response times and end-to-end latencies will have equal values. On the other hand, these values are not the same for the systems modeled with data chains. Therefore, a complete analysis of a real-time system modeled with data chains requires the calculation of not only holistic response times but also end-to-end latencies.

When real-time systems are modeled with both trigger and data chains then end-to-end timing requirements are specified on both types of chains in the component model. These requirements should be unambiguously translated into the analysis model which is required by the analysis tools (implementing end-to-end timing analysis). In such systems, the translation of modeled timing requirements and corresponding timing information into a analysis model is challenging due to several issues.

The first issue is the identification of each individual chain with respect to its type from the modeled application. This issue becomes more challenging in the case of mixed-type chains, i.e., when some tasks in the chain are activated by a single trigger while others are activated by independent triggers as shown in Figure 6. The end-to-end timing requirements in such task chains can correspond to both end-to-end response times and latencies. Another related issue arises when a task chain mimics as a data chain as well as a trigger chain by means of trigger merges as shown in Figure 7. A similar ambiguity exists in the extraction of distributed transactions that contain mixed messages [11], [12] in the network in distributed real-time systems.

Not only such chains should be unambiguously identified, their end-to-end timing requirements should also be translated to the end-to-end timing analysis model. Another issue is to extract the tracing information in each chain (from initiator to the terminator). This can be challenging in the case of distributed real-time systems because a distributed transaction may comprise of a data chain in one node and trigger chain in another while these chains communicate via network messages. Finally, the network timing and message-related information should also be extracted when data chains are distributed over several nodes.

We proposed a method to trace trigger chains in component-based distributed real-time systems in [21]. A method is needed for the identification, tracing, and extraction of data chains from component-based real-time systems; and unambiguous translations of their end-to-end timing requirements into the timing analysis model.

Now, we discuss what do we mean by end-to-end timing requirements in data and trigger chains.

B. End-to-end timing requirements in data chains

A single-node real-time system modeled with three SWCs in RCM is shown in Figure 1. 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.

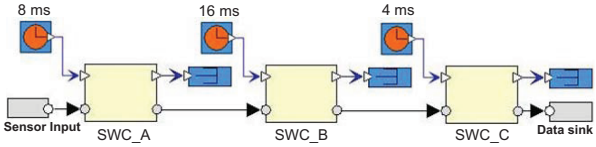

Figure 1. RCM model of a data chain in a single-node real-time system

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 that there are multiple outputs corresponding to a single input signal. The four end-to-end latency semantics are identified in Figure 2.

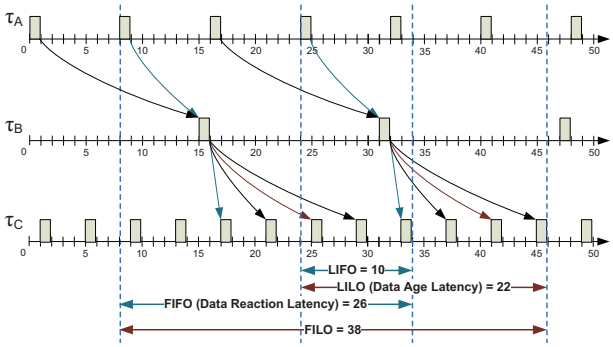

Figure 2. End-to-end latencies of a data chain in a real-time system

1) 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).

2) 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 [16]. 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.

3) 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 [16]. Data reaction is 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 input is important.

4) 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.

The data chains may also be distributed over more than one nodes in distributed real-time systems. Consider a model of a two-node distributed real-time 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. 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 the single-node real-time system example discussed above, it can be scheduled in a similar manner as indicated by τ_A, τ_B and τ_C in Figure 2. The end-to-end latencies are also defined in a similar fashion.

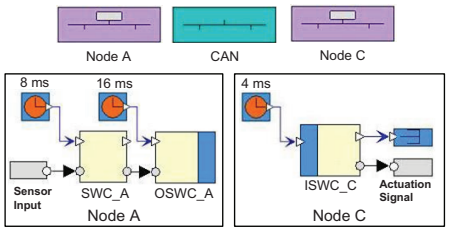

Figure 3. RCM model of a data chain in a distributed real-time system

C. End-to-end timing requirements in trigger chains

An example of a trigger chain that consists of three components is shown in Figure 4. Assume that each components corresponds to a task at run-time. When task τ_{SWC_A} finishes its execution, it triggers τ_{SWC_B}. Similarly, τ_{SWC_C} can only be triggered by τ_{SWC_B} after finishing its execution. There cannot be multiple outputs corresponding to a single input signal. In fact, there will always be one output of the chain corresponding to the input trigger. The focus in a trigger chain is on the calculation of the holistic response-response time only. Hence, the end-to-end timing requirements correspond to the holistic response times. In order to provide a comparison of holistic response time in a trigger chain with the end-to-end latencies in a data chain, assume that the trigger chain shown in Figure 4 is the only chain of tasks in the system. Let the priorities of all tasks be the same while WCET of each task is 1ms. The holistic response time of this trigger chain is equal to the response time of τ_{SWC_C} which is, intuitively, equal to 3ms.

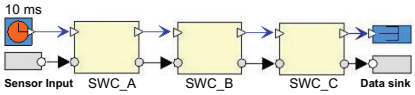

Figure 4. RCM model of trigger chain in a single-node real-time system

Distributed real-time systems can also be modeled with trigger chains. Consider a model of a two-node distributed real-time system modeled with RCM as shown in Figure 5. There is only one triggering ancestor in node A that activates SWC_A. The $ISWC\_C$ in only activated when an interrupt is raised due to the arrival of a CAN message at node C. Once again, the end-to-end timing requirements correspond to end-to-end response times.

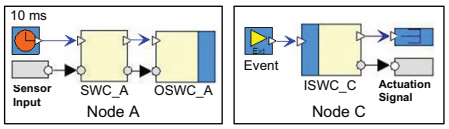

Figure 5. RCM model of trigger chain in a distributed real-time system

IV. Guidelines for the Solution

We provide preliminary guidelines for the development of a method to identify, trace and extract data chains from

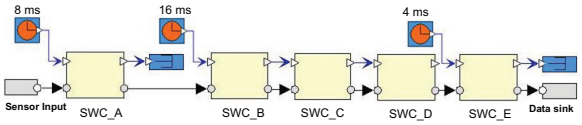

Figure 6. RCM model of a mixed-type chain

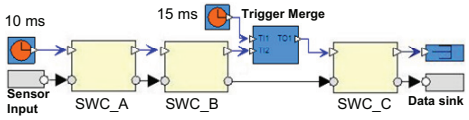

Figure 7. RCM model of a data chain containing trigger merges

component-based real-time systems. The method will also support unambiguous translations of the end-to-end timing requirements specified on data and trigger chains into the analysis model. The new method will be adapted from the existing method in Rubus-ICE [21] to extract the complete tracing information from all data chains.

We will introduce a new object in the component model called trigger map that will extract the triggering information for each task in every chain. Based on this information in the trigger map, an iterative method will determine whether the triggering of every two neighboring tasks in a chain is dependent or independent of each other. If all the triggers are dependent on the initial trigger then the chain will be identified as a trigger chain. If there exists at least one trigger, in the signal map of a chain, that is independent of the rest then the chain will be identified as a data chain. If trigger merges are identified in the trigger map of a chain, it will be regarded as the a data chain. This method will be iterated for all the chains in the system.

The new method will translate the extracted timing information and the trigger map to the analysis model that will input to the analysis tools in XML format. Based on this model, the analysis tools will perform end-to-end response-time and latency analysis. When this method is fully developed, we will implement it in Rubus-ICE as a proof of concept. We believe, this solution will also be applicable to several other component models for real-time systems that use a pipe-and-filter style for component interconnection, e.g., ProCom [22].

V. Summary

We discussed the problem concerning the issues that arise when end-to-end timing requirements are translated into the analysis model from component-based real-time systems that are modeled with data and trigger chains. The end-to-end timing requirements on trigger chains are different from those on data chains. We distinctively identified these requirements in data and trigger chains within single-node and distributed real-time systems. These timing requirements should be unambiguously translated into the analysis model which serves as an input to the analysis tools integrated with the component model. We provided preliminary guidelines for the development of a method to identify, trace and extract data chains and unambiguous translations of their end-to-end timing requirements into a analysis model. Currently, we are developing this method and, in parallel, implementing the end-to-end latency analysis in Rubus-ICE. We plan to provide a proof of concept by conducting an industrial case study using Rubus-ICE.

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, Sweden.

References

[1] 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.

[2] 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*, vol. 28, no. 2/3, pp. 101–155, 2004.

[3] M. Nolin, J. Mäki-Turja, and K. Hänninen, "Achieving Industrial Strength Timing Predictions of Embedded System Behavior," in *ESA*, 2008, pp. 173–178.

[4] K. W. Tindell, "Using offset information to analyse static priority preemptively scheduled task sets," Dept. of Computer Science, University of York, Tech. Rep. YCS 182, 1992.

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Worst-case response-time analysis for mixed messages with offsets in controller area network," in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2012.

[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] 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*.

[16] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics," in *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, dec. 2008.

[17] 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*, ser. EMSOFT '10. ACM, 2010, pp. 129–138.

[18] "Arcticus Systems," http://www.arcticus-systems.com.

[19] 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.

[20] 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)*, Sep. 2011, pp. 229–238.

[21] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extraction of end-to-end timing model from component-based distributed real-time embedded systems," in *Time Analysis and Model-Based Design, from Functional Models to Distributed Deployments (TiMoBD) workshop*. Springer, October 2011, pp. 1–6.

[22] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A Component Model for Control-Intensive Distributed Embedded Systems," in *Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE2008)*.