Real-Time Analysis of Distributed Systems including Tasks with Variable Rate-dependent Behavior

Timo Feld, Uwe Werkmann, Frank Slomka
Institute of Embedded Systems/Real-Time Systems
Ulm University, Germany
firstname.lastname@uni-ulm.de

Abstract. In automotive production car engines there are tasks, where the frequency of activation and execution times vary with the angular velocity of the engine. The variable rate-dependent behavior (VRB) task model has been proposed as a means of modeling this behavior. In the literature a variety of real-time analysis have been developed for tasks with this dependency of an angular velocity. However, these analysis focus on mono-processor systems. In this paper we propose the first analysis for distributed systems including VRB-tasks.

1. Introduction

On an engine control unit, some tasks depend on the rotation speed of the engine. For instance, the task that calculates the quantity of fuel that should be injected: the higher the engine speed, the more frequent this task is executed and the lower the relative deadline. Furthermore, different algorithms with different execution times are used at different ranges of the engine speed. In summary, the frequency of activation, the deadlines and the execution times depend on the rotation speed of the engine. In the literature these kind of tasks have been given a variety of names. We use the term *task with Variable Rate-dependent Behavior* (VRB-task) [7].

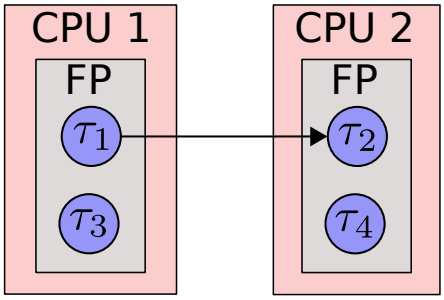


Figure 1: A distributed system

In recent years, a variety of publications focused on real-time analysis for such VRB-tasks. All these works focus on mono-processor systems. However, today's embedded systems are increasingly based on multiprocessors for higher performance and lower power consumption. One type is the distributed system, which is particular challenging regarding the real-time analysis. In Figure 6 a simple distributed system with two CPUs is illustrated. As shown task τ_1 triggers task τ_2 and because CPU1 effects CPU2, this system cannot be analyzed as two mono-processor

systems. The Real-Time Calculus (RTC)-toolbox is a powerful Matlab-toolbox for the analysis of such distributed systems. In order to analyze distributed systems that contain VRB-tasks, we provide in this paper methods to integrate VRB-tasks in the RTC-toolbox.

1.1. Related Work

The problem of scheduling VRB-tasks has been addressed by Kim et al. [13] (referred to as rhythmic tasks). They propose an analysis under very restrictive assumptions with exactly one rhythmic task. In Pollex et al. [15] [16] a sufficient analysis for a task set with multiple VRB-tasks (referred to as engine-triggered tasks) has been described. The accuracy is improved by Feld and Slomka [8] by additionally considering angular phases between tasks. Davis et al. [7] propose analyses for VRB-tasks using integer linear programming (ILP) to find the maximum interference. Huang and Chen [11] describe a sufficient schedulability test for VRB-tasks under fixed priority mode-level scheduling, where the same task can have different priorities for each mode. Buttazzo et al. [6], Guo and Baruah [10], Biondi and Buttazzo [2] and Biondi et al. [3] present analyses for Earliest Deadline First (EDF)-scheduling. Mohaqeqi et al. [14] propose a representation of a rate-dependent task with the Digraph Real-time task model and show that their representation results in a safe analysis. In [4], Biondi et al. presented an exact characterization of interference. Based on this method, Biondi et al. [5] introduced exact schedulability tests for FP-scheduling. Most recently Feld and Slomka [9] proposed a method to characterize the exact interference of a VRB-task. In contrast to [4] they account for arbitrary but bounded accelerations and reduce the runtime complexity.

All these methods are for uniprocessor systems. In [12], Huang and Chen present utilization bounds for multiprocessor systems with partitioned scheduling, where each task is statically assigned to one processor. Note that such a multiprocessor system differs from a distributed system, since a task does not effect a task of another processor. To the best of our knowledge there is no analysis for distributed systems that contain VRB-tasks.

2. Computational Model

In this section, we describe the VRB-task model and the general system model.

2.1. VRB-task model

A VRB-task is triggered according to the rotation of the engine (or in general a rotating source). The number of rotations accomplished is denoted by φ. The angular velocity is denoted by ω and bounded in $[\omega^-, \omega^+]$. ω is the first derivation of φ:

$$\omega = \dot{\varphi} = \frac{d\varphi}{dt} \quad \left[\frac{1}{s}\right] \tag{1}$$

The acceleration is denoted by a and bounded in $[a^-, a^+]$[1]. It is the second derivation of φ:

$$a = \dot{\omega} = \ddot{\varphi} = \frac{d^2\varphi}{dt^2} \quad \left[\frac{1}{s^2}\right] \tag{2}$$

[1]Note that in section 4.3 we assume a constant acceleration of $a = 0$.

For convenience we assume that ω (including ω^- and ω^+) is measured in revolutions per second and the acceleration a in revolutions per second squared.

Inter-arrival time: The inter-arrival time T is the distance in time between two successive events. For a VRB-task the inter-arrival time is stated in terms of a fraction of a full rotation. This fraction is denoted by b. For instance $b = 1$ if the inter-arrival time equals a full rotation. Since the angular velocity is given in revolutions per second, the inter-arrival time T for a constant speed ω can be determined with $T = \frac{b}{\omega}$. The minimum and maximum inter-arrival times are therefore: $T_{min} = \frac{b}{\omega^+}$ and $T_{max} = \frac{b}{\omega^-}$.

Execution time: There are several execution modes. The number of modes is denoted by M and the index of an execution mode by m. The execution time of a VRB-task depends on the previous inter-arrival time T according to equation (3):

$$C(T) = \begin{cases} C_1 & \text{if } T_{max} \geq T \geq T_1 \\ C_2 & \text{if } T_1 > T \geq T_2 \\ \dots & \\ C_M & \text{if } T_{M-1} > T \geq T_M \end{cases} \tag{3}$$

T_1 is the smallest inter-arrival time for the execution mode 1, T_2 for mode 2 and so on. The smallest inter-arrival time for mode M (T_M) equals the smallest inter-arrival time of the system: $T_M = T_{min}$. The largest inter-arrival time for execution mode 1 is denoted by T_1^u, for mode 2 by T_2^u and so on. According to equation 3 it holds $T_1^u = T_{max}$, $T_2^u = T_1$ and so on. Each execution mode m is thus characterized by (C_m, T_m, T_m^u) representing the worst-case execution time C_m (with $C_1 \geq C_2 \geq \ldots \geq C_M$), minimum inter-arrival time T_m and maximum inter-arrival time T_m^u for a job executing in that mode.

We note that, in the VRB-task model, the *average* angular velocity (i.e. the inter-arrival time) causes the execution mode and accelerations are arbitrary but bounded. In contrast the AVR-task model [4] assumes accelerations to be fixed between two events and it assumes that the *instantaneous* speed causes the execution mode.

2.2. System model

The system that is being analyzed has two or more resources each executing a set of tasks Γ. Each task may be triggered according to the sporadic task model with a fixed inter-arrival time T and a fixed net execution time C, by a rotating source as described in section 2.1 or by the outgoing events of another task.

We note that a task (whether VRB or sporadic task) is typically also defined by its deadline. However, since the provided methods in this paper do not depend on the deadline, we do not state these explicitly.

3. Recapitulation of existing real-time analysis

3.1. Exact request bound function

The *request bound function* $\alpha^u(\Delta)$ expresses the maximum accumulated execution time that a task can require from the processor within any time interval Δ of given length. In [9] a method is proposed which determines the exact request bound function (referred to as the worst-case interference) generated by one VRB-task in a given interval Δ. The general approach is to identify all courses of the angular velocity (referred to as paths) that can lead to the highest execution

requirement in a given interval Δ. The number of courses that has to be considered increases exponentially with increasing interval length. However, for a limited interval Δ the result of this method is an exact request bound function $\alpha^u_{exact}(\Delta)$. Due to space limitations we refer to [9] for more details.

3.2. Real-time Calculus

The Real-Time Calculus (RTC)-toolbox is a powerful Matlab-toolbox, which allows to analyze distributed real-time systems. It represents each task with the Greedy Processing Component (GPC) and describes the stimulation of the system with arrival- and service-curves. Figure 2 shows a GPC.

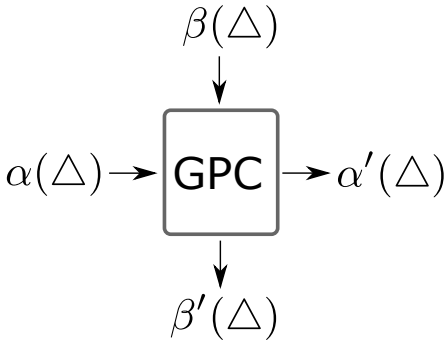
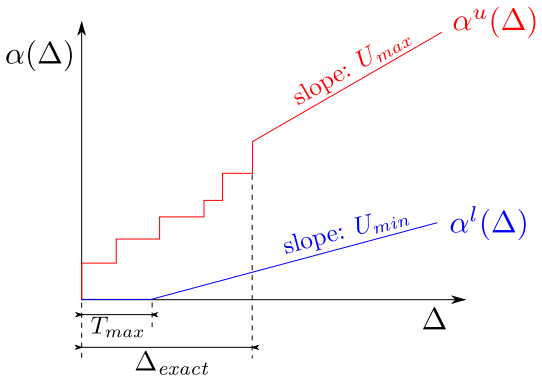


Figure 2: Greedy Processing Component

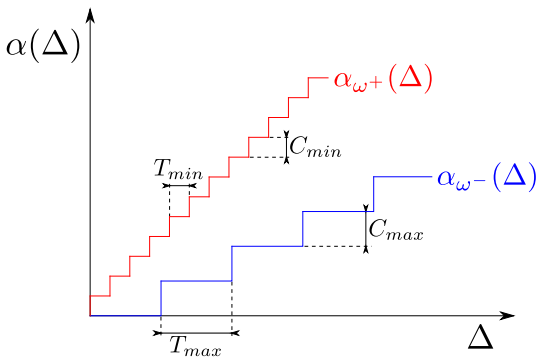
The service curve $\beta(\Delta)$ describes how long the processor is available in a given time interval Δ. The arrival curve $\alpha(\Delta)$ describes the number of events that require processor resources. In combination with the execution times of those events it describes the accumulated requested execution time in a given time interval Δ and is identical to the request bound function (see $\alpha^u_{exact}(\Delta)$ in section 3.1). Given both incoming curves, the outgoing curves can be determined: the remaining service curve $\beta'(\Delta)$ and the outgoing event density $\alpha'(\Delta)$. The remaining service curve describes how much resources are available for remaining tasks. The outgoing event density describes the number of events that are completed in a given time interval. This outgoing event density is of relevance if these events trigger a different task (for instance of another CPU). Each of these curves have an upper and lower version denoted with ^{u} and ^{l}. Intuitively the subtraction of the requested execution time $\alpha^u(\Delta)$ from the available processor time $\beta^l(\Delta)$ is the remaining processor time (i.e. the remaining service curve $\beta'^l(\Delta)$). Similarly the outgoing event density depends on the number of incoming events and is bounded by the available processor time. The determination of the outgoing event density and remaining service curve uses the operations: convolution ($\otimes$) and deconvolution ($\oslash$) in Min ($\otimes, \oslash$) and Max-Plus-Algebra ($\overline{\otimes}, \overline{\oslash}$). We recap the respective complete equations in (4). Due to page limitations we refer to [17] for a more detailed explanation.

$$\begin{aligned} \alpha'^l &= \min((\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l) \\ \alpha'^u &= \min((\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u) \\ \beta'^l(\Delta) &= (\beta^l(\Delta) - \alpha^u(\Delta)) \overline{\otimes} 0 \\ \beta'^u(\Delta) &= (\beta^u(\Delta) - \alpha^l(\Delta)) \overline{\oslash} 0 \end{aligned} \tag{4}$$

Given these curves, the maximum response time (also referred to as the delay) of each task



(a) $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$



(b) $\alpha_{\omega^+}(\Delta)$ and $\alpha_{\omega^-}(\Delta)$

Figure 3: Requested execution time

is determined. The schedulability test is conducted by assuring that the delay is lower than the corresponding deadline of each task.

4. Real-Time Analysis

In this section, we present an analysis for distributed systems with VRB-tasks. This analysis uses the RTC-toolbox. In order to include a VRB-task in the RTC-toolbox and conduct the schedulability test of such a distributed system three methods need to be adapted for VRB-tasks:

- the requested execution time of a VRB-task,
- the outgoing event density of a VRB-task
- and the remaining service curve of a VRB-task.

In the next subsections, we address these methods.

4.1. Requested execution time

We distinguish in the upper and lower bound of the requested execution time.

4.1.1. Upper bound on requested execution time $\alpha^u(\Delta)$

With the method described in [9] the maximum requested execution time of a VRB-task can be determined. We depict this method with $\alpha^u_{exact}(\Delta)$ (see section 3.1). However, the RTC-toolbox requires its curves to be defined for unlimited intervals[2]. We therefore construct this curve such that the first part is exactly computed with $\alpha^u_{exact}(\Delta)$ according to [9] and after a certain interval length it is approximated with a linear upper bound (this curve is illustrated in figure 3 a). Note that the determination of the exact requested execution time of a VRB-task for unlimited intervals we leave as a future work.
Linear upper bound

[2] A curve can be defined with the method *rtccurve*, which uses periodic elements to describe the curve for unlimited time intervals[1].

The utilization is the increase in execution requirement in relation to the interval length. This utilization of one mode is computed with $U_m = \frac{C_m}{T_m}$. The maximum utilization of a VRB-task is thus:

$$U_{max} = \max_{m \in \mathbb{N} | m \leq M} \left(\frac{C_m}{T_m} \right). \tag{5}$$

This U_{max} is the slope of the upper bound. The maximum requested execution requirement for an interval $\Delta = 0$ is bounded by the maximum execution time C_{max}, since at most one event can occur given the stimulation of a VRB-task. The maximum starting point and the maximum slope result in the following linear upper bound:

$$\alpha^u_{linear}(\Delta) = C_{max} + U_{max}\Delta \tag{6}$$

For the first interval Δ_{exact}, the method $\alpha^u_{exact}(\Delta)$ is used to determine the exact requested execution time and for any larger interval the linear upper bound is taken. This results in equation 7:

$$\alpha^u(\Delta) = \begin{cases} \alpha^u_{exact}(\Delta) & \text{, if } \Delta < \Delta_{exact} \\ C_{max} + U_{max}\Delta & \text{, if } \Delta \geq \Delta_{exact} \end{cases} \tag{7}$$

A larger Δ_{exact} results in a more precise analysis. But this comes with the cost of analysis runtime, as the runtime of the exact method increases exponentially with increasing interval length [9].

4.1.2. Lower bound on requested execution time $\alpha^l(\Delta)$

Analogously to the upper bound, the minimum increase in execution time in relation to interval length is the minimum utilization. With the maximum inter-arrival time T^u_m of mode m the minimum utilization is:

$$U_{min} = \min_{m \in \mathbb{N} | m \leq M} \left(\frac{C_m}{T^u_m} \right) \tag{8}$$

The maximum inter-arrival time T_{max} of a VRB-task is the longest time in which no event can occur. Hence for any interval $\Delta > T_{max}$ there is an execution requirement. With this latest starting point and the minimum slope we conclude the lower bound on the execution requirement with equation 9 (this curve is illustrated in figure 3 a):

$$\alpha^l(\Delta) = \max((\Delta - T_{max})U_{min}, 0) \tag{9}$$

4.2. Remaining service curve

We restate from equations 4 in chapter 3.2, the determination of the remaining service curve:

$$\begin{aligned} \beta'^l(\Delta) &= (\beta^l(\Delta) - \alpha^u(\Delta)) \overline{\otimes} 0 \\ \beta'^u(\Delta) &= (\beta^u(\Delta) - \alpha^l(\Delta)) \overline{\oslash} 0 \end{aligned} \tag{10}$$

The available service curves $\beta^l(\Delta)$ and $\beta^u(\Delta)$ are independent of a rotational source (i.e. are not computed differently for VRB-tasks). $\alpha^l(\Delta)$ and $\alpha^u(\Delta)$ for VRB-tasks are provided with the previous subsection. Hence by inserting equations 7 and 9 into equations 10 the remaining service curve can be determined.

4.3. Outgoing event density

The aim is to find the maximum number of processed events for any given time interval Δ. For simplification we assume steady state conditions (with $a = 0$) in this section. The computation for arbitrary accelerations we leave as a future work.

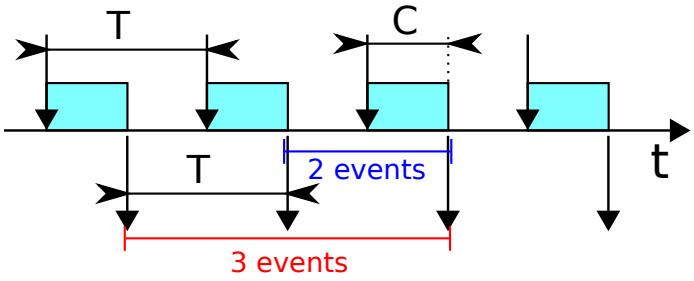


Figure 4: Outgoing event density

Figure 4 shows a simple example of one task that has the full processor capacity available. In this example events occur with an inter-arrival time of T each having an execution time of C. Since full processor capacity is assumed in this example, always C after the occurring of each event it is processed. The processing of an event (i.e. the outgoing event) is indicated with a down-arrow below the time-line. Figure 4 shows an interval length which includes two outgoing events and another interval that includes three outgoing events. These show that the distance between each two outgoing events equals the inter-arrival time. It follows:

1. A smaller inter-arrival time leads to a higher number of processed events in an interval Δ.

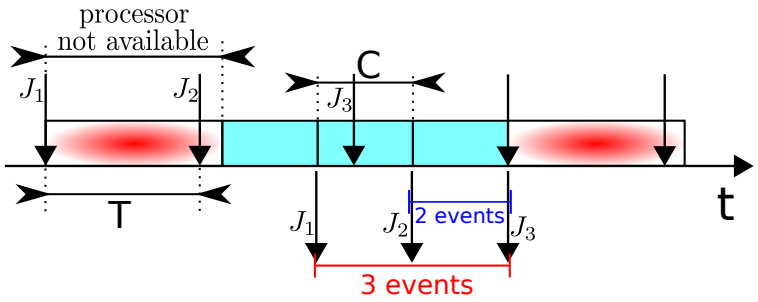


Figure 5: Outgoing event density with limited $\beta^u(\Delta)$

In figure 5 we illustrate an example where the processor is not fully available. According to the available service curve $\beta^u(\Delta)$ the processor is not available during certain times as labeled in the figure. The occurring and processing of jobs are denoted with J_1, J_2 and J_3. Figure 5 illustrates that the length of the execution time limits the distance between outgoing events. It follows:

2. The distance between each two outgoing events is not less than the execution time. Hence a lower execution time leads to a higher (or at least the same) number of processed events.

From the computational model (see section 2.1) it follows:

3. The highest angular velocity causes the smallest inter-arrival time.

4. The execution times are decreasing with increasing angular velocity. Hence the smallest execution time is at the highest angular velocity.

From points 1. to 4. it follows: the highest angular velocity ω^+ produces the maximum outgoing event density $\alpha'^u(\Delta)$. Therefore the incoming curve $\alpha_{\omega^+}(\Delta)$ for this purpose is constructed with $T_{min} = b/\omega^+$ and $C_{min} = C(T_{min})$:

$$\alpha_{\omega^+}(\Delta) = \left\lceil \frac{\Delta}{T_{min}} \right\rceil C_{min} \tag{11}$$

Analogously the minimum outgoing event density is produced at the lowest angular velocity. Hence the respective incoming curve is constructed with the highest inter-arrival time $T_{max} = b/\omega^-$ and the largest execution time $C_{max} = C(T_{max})$:

$$\alpha_{\omega^-}(\Delta) = \left\lfloor \frac{\Delta}{T_{max}} \right\rfloor C_{max} \tag{12}$$

Both curves (α_{ω^+} and α_{ω^-}) are illustrated in figure 3 b. We use the determination of the outgoing event density from equations 4 in chapter 3.2 and replace the respective functions with $\alpha_{\omega^+}(\Delta)$ and $\alpha_{\omega^-}(\Delta)$ to adapt for VRB-tasks:

$$\begin{aligned} \alpha'^u &= \min((\alpha_{\omega^+} \otimes \beta^u) \oslash \beta^l, \beta^u) \\ \alpha'^l &= \min((\alpha_{\omega^-} \oslash \beta^u) \otimes \beta^l, \beta^l) \end{aligned} \tag{13}$$

As described in the last subsections (using equations 7, 9, 10 and 13) all relevant methods are modified in order to include VRB-tasks in the RTC-toolbox. Thus to determine the delay of a VRB-task or to determine any necessary curves for the remaining regular tasks (that are not VRB-tasks), the RTC-toolbox can be used. A tutorial and description of these methods provided by the RTC-toolbox is given in [1].

5. Experiments

Using the methods described in section 4 and the RTC-toolbox a distributed system can be analyzed. Another advantage of incorporating VRB-tasks into the RTC-toolbox is that this system can use several schedulers as well as combinations of those. To illustrate the powerfulness of our new method, we implemented the analysis of the distributed system illustrated in figure 6. The system consists of two CPUs connected with a Bus. Several schedulers are used: CPU1 uses Fixed Priority- and CPU2 Earliest Deadline First-scheduling. The bus uses Time Division Multiple Access (TDMA), where tasks τ_5 and τ_7 share one TDMA-slot with fixed priorities.

Tasks τ_1 and τ_4 are VRB-tasks and have the following parameters:

- acceleration is bounded in $a^- = -100\,\frac{1}{sec^2}$ and $a^+ = 100\,\frac{1}{sec^2}$.
- the speed-range is $\omega^- = 1000\,rpm$, $\omega^+ = 5000\,rpm$ with 4 execution modes: [$1000\,rpm$, $2000\,rpm$], ($2000\,rpm$, $3000\,rpm$], ($3000\,rpm$, $4000\,rpm$], ($4000\,rpm$, $5000\,rpm$].
- execution times (of each execution mode) of task τ_1 are $C_1 = 2.4\,ms$, $C_2 = 2\,ms$, $C_3 = 1.35\,ms$ and $C_4 = 0.9\,ms$
- execution times of task τ_4 are $C_1 = 4.2\,ms$, $C_2 = 3\,ms$, $C_3 = 2.5\,ms$ and $C_4 = 1.86\,ms$

- the rotation scaling factor is for both tasks $b = 1$, i.e. an event occurs every full rotation.

Task τ_9 is a sporadic task and has the parameters $T = 40\,ms$ and $C = 8\,ms$. The other tasks are stimulated by the outgoing event density of the preceding tasks. The main result of this analysis is:

- The complete delay of the path stimulated by S1 is:
  - $d(S1) = d(\tau_1) + d(\tau_2) + d(\tau_3) = 3 + 9 + 8 = 20\,ms$
- The complete delay of the path stimulated by S2 is:
  - $d(S2) = d(\tau_4) + d(\tau_5) + d(\tau_6) + d(\tau_7) + d(\tau_8) = 7.4 + 3 + 5 + 4.5 + 15 = 34.9\,ms$
- The complete delay of the path stimulated by S3 is: $d(S3) = d(\tau_9) = 14.6\,ms$

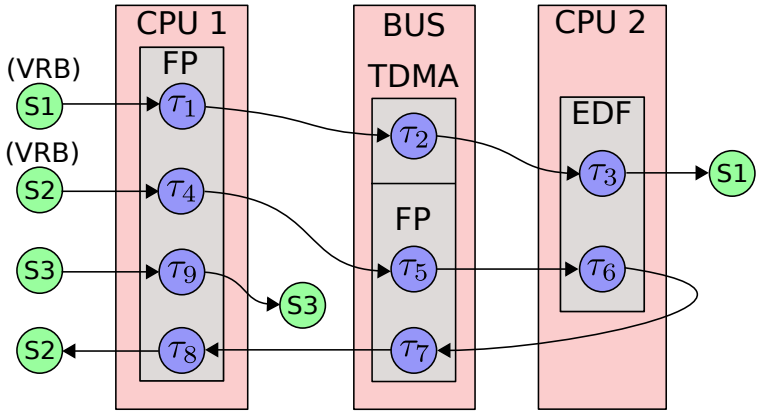


Figure 6: Distributed system

6. Summary and Future Work

In this paper we presented the first analysis for distributed systems with VRB-tasks. To illustrate the powerfulness of the method, we analyzed a system that uses a mix of Fixed Priority, Earliest Deadline First and TDMA-scheduling. To deal with the requirement of curves being defined for unlimited intervals, we used an upper bound on the request bound function. In a future work, we want to determine the exact request bound function for unlimited intervals. Hence, having precise knowledge of the execution requirement for arbitrary high time intervals and thus avoiding the need for an approximation.

References

[1] *Real-time calculus toolbox tutorial*. `http://www.mpa.ethz.ch/static/Tutorial.html`.

[2] Biondi, Alessandro and Giorgio Buttazzo: *Engine control: task modeling and analysis*. In *Design Automation and Test in Europe (DATE)*, 2015.

[3] Biondi, Alessandro, Giorgio Buttazzo, and Stefano Simoncelli: *Feasibility analysis of engine control tasks under EDF scheduling*. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2015.

[4] Biondi, Alessandro, Alessandra Melani, Mauro Marinoni, Marco Di Natale, and Giorgio Buttazzo: *Exact interference of adaptive variable-rate tasks under fixed-priority scheduling*. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2014.

[5] Biondi, Alessandro, Marco Di Natale, and Giorgio Buttazzo: *Response-time analysis for real-time tasks in engine control applications*. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems (ICCPS)*, pages 120–129, 2015.

[6] Buttazzo, G., E. Bini, and D. Buttle: *Rate-adaptive tasks: Model, analysis and design-issues*. In *Design Automation and Test in Europe (DATE)*, 2014.

[7] Davis, R., T. Feld, V. Pollex, and F. Slomka: *Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling*. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.

[8] Feld, Timo and Frank Slomka: *Sufficient response time analysis considering dependencies between rate-dependent tasks*. In *Design Automation and Test in Europe (DATE)*, 2015.

[9] Feld, Timo and Frank Slomka: *Exact interference of tasks with variable rate-dependent behaviour*. accepted at Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2017. Digital Object Identifier: 10.1109/TCAD.2017.2729459.

[10] Guo, Zhishan and Sanjoy Baruah: *Uniprocessor EDF scheduling of AVR task systems*. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems (ICCPS)*, pages 159–168, 2015.

[11] Huang, Wen Hung and Jian Jia Chen: *Techniques for schedulability analysis in mode change systems under fixed-priority scheduling*. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2015.

[12] Huang, Wen Hung and Jian Jia Chen: *Utilization bounds on allocating rate-monotonic scheduled multi-mode tasks on multiprocessor systems*. In *Design Automation Conference (DAC)*, 2016.

[13] Kim, Junsung, Karthik Lakshmanan, and Ragunathan Rajkumar: *Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems*. In *Proceedings of the IEEE/ACM Third International Conference on Cyber-Physical Systems (ICCPS)*, pages 55–64, 2012.

[14] Mohaqeqi, Morteza, Jokaria Abdullah, Pontus Ekberg, and Wang Yi: *Refinement of workload models for engine control by state space partitioning*. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2017.

[15] Pollex, Victor, Timo Feld, Frank Slomka, Ulrich Margull, Ralph Mader, and Gerhard Wirrer: *Sufficient real-time analysis for an engine control unit*. In *Proceedings of the 21st International conference on Real-Time Networks and Systems (RTNS)*, pages 247–254, 2013.

[16] Pollex, Victor, Timo Feld, Frank Slomka, Ulrich Margull, Ralph Mader, and Gerhard Wirrer: *Sufficient real-time analysis for an engine control unit with constant angular velocities*. In *Design Automation and Test in Europe (DATE)*, 2013.

[17] Wandeler, E.: *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, September 2006. Chapter 2.