A Scalable Approach for the Description of Dependencies in Hard Real-Time Systems*

Steffen Kollmann, Victor Pollex, Kilian Kempf and Frank Slomka

Ulm University
Institute of Embedded Systems/Real-Time Systems
{firstname.lastname}@uni-ulm.de

Abstract. During the design iterations of embedded systems, the schedulability analysis is an important method to verify whether the real-time constraints are satisfied. In order to achieve a wide acceptance in industrial companies, the analysis must be as accurate as possible and as fast as possible. The system context of the tasks has to be considered in order to accomplish an exact analysis. As this leads to longer analysis times, there is a tradeoff between accuracy and runtime. This paper introduces a general approach for the description of dependencies between tasks in distributed hard real-time systems that is able to scale the exactness and the runtime of the analysis. We will show how this concept can be applied to a real automotive application.

1 Introduction

In the last years, designing embedded systems has become a great challenge because more and more applications have to be covered by one system. As a special issue of this development, many new features are only realizable by the connection of different controllers. New cars, for example, have up to 75 ECUs (electronic control units) connected by several buses, which leads to a highly networked system with many requirements, like energy consumption, space or timing behavior.

One specific challenge for the design process are the hard real-time constraints of many applications such as an engine management system or an ABS (anti-lock braking system). Therefore it is necessary to verify in each design iteration whether the real-time constraints of single applications are satisfied or not. A schedulability analysis can be performed in order to determine the worst-case response times of the tasks in a system. To achieve a wide acceptance of such real-time analysis techniques by industrial software developers, tight bounds of the real worst-case response times of the tasks and short runtimes of the analysis are important.

A possibility to get tight bounds is the consideration of dependencies of chained tasksets. Previous work considers different kinds of dependencies, like mutual exclusion of tasks, offsets between task stimulation, task chains or tasks competing for the same resource. But the integration of many other types of dependencies is still unsolved. Especially a holistic model as a new abstraction layer for dependencies is missing. Approximation techniques can be used to achieve low runtimes of the analysis. But this is in conflict to tight worst-case response time bounds.

* This work is supported in part by the Carl Zeiss Foundation.

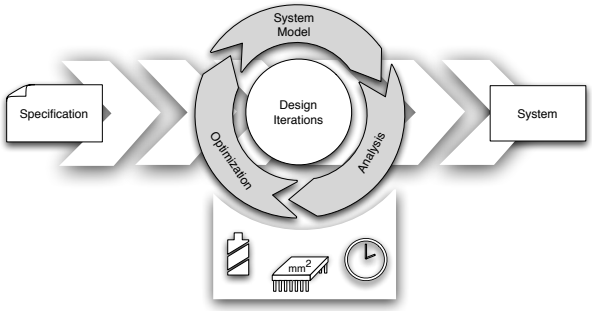

Fig. 1. Design flow of embedded systems

The integration of such real-time analysis techniques into a design process is depicted in figure 1. Starting at the specification, the system model is extracted. The system architecture is then analyzed regarding the different requirements and it is optimized based on the verification results. In early design iterations many different solutions should be considered and therefore it is necessary to have fast verification algorithms. This can be achieved by approximation techniques. In each following iteration the system model is refined and the possible design solutions are bounded. So more and more information about the architecture is available and the verification can work more accurately. While unfortunately this results in longer analysis runtimes, the runtime increase is reasonable because less design solutions have to be considered in consecutive iterations. So it is necessary to have a scalable model which can exploit this tradeoff between runtime and accuracy.

We present such a new holistic model that integrates different types of dependencies into real-time analysis and enables to adjust the level of detail. We show how this general model can be integrated into the schedulability analysis of fixed-priority systems and outline how this approach can be used to approximate the consideration of dependencies in the system and therefore improve the analysis time.

This paper is organized as follows: In section 2 an overview of the related work is given. The model and the corresponding real-time analysis are defined in section 3. Section 4 defines the limiting event stream model and shows how it can be used to describe dependencies in a system in a scalable way. The impact of the introduced analysis is illustrated by a real automotive case-study. The work closes with a conclusion.

2 Related Work

The real-time analysis for distributed systems was introduced by Tindell and Clark [16]. In this holistic schedulability analysis, tasks are considered as independent. This idea has been improved by the transaction model [15], which allows the description of static offsets between tasks. Gutierrez et al. [3] extended this work to dynamic offsets so that the offset can vary from one job of a task to another. Furthermore they have introduced an idea about mutual exclusion of tasks [4] which bases on offsets between tasks. Since

Gutierrez et al. have considered simple task chains, Redell has enhanced the idea to tree-shaped dependencies [13] and Pellizoni et al. applied the transaction model to earliest deadline first scheduling in [12].

Henia et al. used the SymTA/S approach [14] to extend the idea of the transaction model in order to introduce timing-correlations between tasks in parallel paths in distributed systems [5]. This idea has then been improved in [6]. Recently, Rox et al. [7] have described a correlation between tasks caused by an non-preemptive scheduler.

A scalable and modular approach to analyse real-time systems is the real-time calculus (RTC) as presented by Wandeler in [17]. Unfortunately it is not possible to describe dependencies like offsets or mutual exclusion of tasks with the RTC.

Because of the lack of generality or exactness the RTC and the SymTA/S approach are combined in [10], but the authors do not consider any kind of dependency. Another disadvantage of the combination of the models is that the transformation from one model into the other model leads to inaccuracy.

3 Real-Time Analysis

3.1 Model of Computation

In this section we introduce the model necessary for the real-time analysis discussed in section 3.2.

Task Model Γ is a set of tasks mapped onto the same resource $\Gamma = \{\tau_1, ..., \tau_n\}$. A task is a tuple $\tau = (c^+, c^-, b, d, \rho, \Theta^+, \dot{\Theta}^+)$ consisting of : c^+ the worst-case execution time, c^- the best-case execution time, b the blocking time, d the relative deadline of the task, ρ the priority of the task, Θ^+ the maximum incoming stimulation and $\dot{\Theta}^+$ the maximum outgoing stimulation.

Let τ_{ij} be the j-th job/execution of task τ_i. We assume that each job of a task generates an event at the end of its execution to notify other tasks. Furthermore we define $\Gamma_{hp,\tau}$ as a taskset including only tasks having a higher priority than task τ.

Event Model Event streams have been first defined in [2]. Their purpose is to provide a generalized description for any kind of stimulation. The basic idea is to define an event function $\eta(\Delta t, \Theta^+)$ which can calculate for every interval Δt the maximum number of events occurring within Δt (when speaking of an interval, we mean the length of the interval). The event function needs a properly described model behind it which makes it easy to extract the information.

The idea of the maximum event streams is to note for each number of events the minimum interval which can include this number of events. Therefore we get an interval for one event, two events and so on. The interval for one event is infinitely small and therefore considered to be zero. The result is a sequence of intervals showing a non-decreasing behavior. This is because the minimum interval for n events cannot be smaller than the minimum interval for $n-1$ events, as the first interval also includes $n-1$ events.

Definition 1. *An event stream is a set of event stream elements* $\theta : \Theta^+ = \{\theta_1, \theta_2, ..., \theta_n\}$ *and each event stream element* $\theta = (p, a)$ *consists of an offset-interval a and a period p. The event stream complies to the characteristic of sub-additivity:* $\eta(\Delta t_1 + \Delta t_2, \Theta^+) \leq \eta(\Delta t_1, \Theta^+) + \eta(\Delta t_2, \Theta^+)$ *and is monotonically increasing:* $\forall \Delta t_1, \Delta t_2 : \Delta t_1 \leq \Delta t_2 \Rightarrow \eta(\Delta t_1, \Theta^+) \leq \eta(\Delta t_2, \Theta^+)$

Each event stream element θ describes a set of intervals $\{a_\theta + k \cdot p_\theta | k \in \mathbb{N}_0\}$ of the sequence. With an infinite (∞) period it is possible to define a single interval in order to model irregular behavior. Event tuples having infinity as period are called aperiodic elements and event tuples having a period less than infinity are called periodic elements. The event function is defined as follows:

Definition 2. *The event function denotes for an event stream* Θ^+ *and an interval* Δt *the maximum number of events*[1]*:*

$$\eta(\Delta t, \Theta^+) = \sum_{\substack{\theta \in \Theta^+ \\ a_\theta \leq \Delta t}} \left\lfloor \frac{\Delta t - a_\theta}{p_\theta} \right\rfloor + 1 \tag{1}$$

As pseudo-inverse function we define the interval function which denotes the minimum interval in which a given number of events can occur.

Definition 3. *The interval function denotes for an event stream* Θ^+ *and a number of n events the corresponding minimum interval in which these events can occur:*

$$\Delta t(n, \Theta^+) = \inf_{\Delta t \geq 0} \{\Delta t | \eta(\Delta t, \Theta^+) \geq n\} \tag{2}$$

A detailed definition of the concept and the mathematical foundation of the event streams can be found in [1].

An event pattern can be described by the event stream model in several ways. For an efficient implementation of the approach we normalize the event streams as introduced in [8]. With this normalization, the interval function can be efficiently computed. For a detailed description on how to transform an event stream to a normalized event stream and how the interval function can be described efficiently see [8]. In the following we use definition 4.

Definition 4. *A normalized event stream* $\tilde{\Theta}^+$ *has the form*

$$\{(\infty, a_1), \ldots, (\infty, a_m), (p, a_{m+1}), \ldots, (p, a_n)\} : (1 \leq m \leq n \wedge a_i \leq a_j \Leftrightarrow i \leq j \wedge a_n - a_{m+1} \leq p)$$

which means that the event stream has first an aperiodic part and then a periodic part where each periodic element has the same period and the events occur within the same periods. All elements are sorted by their offsets. We also define that $N^{\infty}_{\tilde{\Theta}^+}$ is the number of aperiodic elements of an event stream and $N^{p}_{\tilde{\Theta}^+}$ is the number of periodic elements of an event stream. With this definition and the methodologies introduced in [8] and [9] an efficient real-time analysis for distributed systems is possible. For the rest of the paper we assume that we use only normalized event streams.

[1] Since in this case study we consider a fixed priority non-preemptive scheduler, the floor operation is used. This is also a bound for preemptive schedulers, but for these, better bounds can be given.

3.2 Holistic Real-Time Analysis

Based on previous work we define the real-time analysis with event streams. As described in [16], in each global iteration step of the real-time analysis the worst-case response time and the outgoing stimulation for each task in the system are computed until a fix-point is found. The real-time analysis with event streams is described in [9]. In that paper the computation of the worst-case response time, the best-case response time, and the outgoing stimulations are given. Since the paper at hand focuses on the improvement of the worst-case response time of a task, only that part of the analysis will be reproduced here.

Based on the worst-case response time analysis in [11] we can define the analysis for event streams as follows:

Lemma 1. *The worst-case response time of a task is bounded by:*

$$r^+(\tau) = \max_{k\in[1,\ldots,m]}\{r^+(k,\tau) - \Delta t(k,\Theta_\tau^+)\}\ m = \min_{k\in\mathbb{N}}\{k | r^+(k,\tau) \leq \Delta t(k+1,\Theta_\tau^+)\}$$
$$r^+(k,\tau) = \min\{\Delta t | \Delta t = b_\tau + k\cdot c_\tau^+ + \sum_{\tau_i\in\Gamma_{hp,\tau}} \eta(\Delta t,\tau_i)\cdot c_{\tau_i}^+\} \quad (3)$$

Proof. The proof is given in [9].

4 Considering Dependencies

The lack of the previously discussed model is that dependencies between stimulations caused by the system context are not considered during the real-time analysis. Therefore we will extend the model introduced in section 3 by the limiting event streams. Via this extension we are able to describe different kinds of dependencies with a single holistic model.

The idea of this new technique is depicted in figure 2. Four tasks are mapped onto a resource that is scheduled non-preemptively with fixed-priorities. For each task we determine the worst-case response time. Assume task τ_4 as the analyzed task. If no correlations between the stimulations are considered the worst-case response time of the task is computed by equation (3). This means the interference produced by the tasks τ_1 to τ_3 is maximal concerning τ_4.

Now assume that the stimulations have an offset dependency to each other. In order to model such a correlation, the stimulations can be considered as vertices in a graph and the correlation between them as edges. For each maximal clique in the graph of size 2 or bigger, a set S_i is build which contains all stimulations that are represented by the nodes of the clique. Note that these cliques are usually given as input and are not being searched for. To consider every possibility which is described by the correlations, the power set $\mathscr{P}(S_i)$ of such a previously mentioned set is taken. For each element $M \in \mathscr{P}(S_i)$ of such a power set a limitation for the number of events that can occur by the combined stimulations in M is computed.

This very modular concept permits to consider correlations in a scalable way by choosing only a subset of the power set which should be considered in the analysis. See the three possibilities in figure 2. In part a) only one limitation over all stimulations is

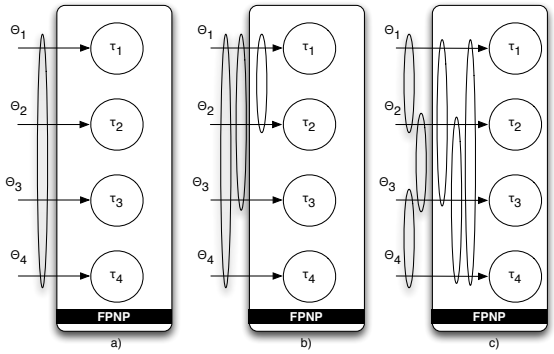

Fig. 2. Possible consideration of relations by limiting event streams

used. Out of all stimulations of the corresponding power set $\mathscr{P}(S_i)$ the second example uses only those that contain exactly the stimulations of the k-highest priority tasks that are mapped on the resource, where $2 \leq k \leq n$ and n is the number of tasks mapped on a resource. The last one c) considers only the pairwise correlations. That way the developer has the chance to determine the level of detail in the analysis. Next we will formulate the limiting event streams formally.

Definition 5. *The limiting event stream is an event stream that defines the maximum occurrence of events for a set of event streams. The limiting event stream is defined as* $\overline{\Theta} = (\Theta^+, \overrightarrow{\Theta})$. Θ^+ *describes the limitation for a set of event streams* $\overrightarrow{\Theta}$. *The limiting event stream fulfills the condition:*

$$\eta(\Delta t, \Theta^+_{\overline{\Theta}}) \leq \sum_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \eta(\Delta t, \Theta)$$

Example 1. If no correlations between event streams are defined, the limiting event stream is definded by: $\overline{\Theta} = (\cup_{\Theta \in \overrightarrow{\Theta}} \Theta, \overrightarrow{\Theta})$.

Example 2. Assume $\Theta^+_A = \Theta^+_B = \{(20,0)\}$ and an offset of 10 t. u. between these two event streams. The cumulated occurrence of events when the offset is not considered is described by $\eta(\Delta t, \Theta^+_A \cup \Theta^+_B)$. The cumulated occurrence considering the offset of the events can be described by the limiting event stream $\overline{\Theta} = (\{(20,0),\ (20,10)\}, \{\Theta_A, \Theta_B\})$. If we consider the event streams as independent, two events occur in an interval $\Delta t = 5$ t. u. $(\eta(\Delta t, \Theta^+_A \cup \Theta^+_B))$. In contrast, in the same interval only one event occurs if the dependency is considered $(\eta(\Delta t, \Theta^+_{\overline{\Theta}}))$.

Next we define how a limiting event stream can be calculated.

Definition 6. *Let* $\Delta\beta : \mathbb{N} \to \mathbb{R}$ *be a limiting interval function which assigns a minimal time interval from a given number of events in subject to a relationship of event streams* $\overrightarrow{\Theta} := \{\Theta_1, \ldots, \Theta_n\}$, *then a limitation for a limiting event stream* $\overline{\Theta}$ *can be determined by:*

$$\Theta^+_{\overline{\Theta}} := \nu(\overrightarrow{\Theta}_{\overline{\Theta}}, \Delta\beta(n))$$

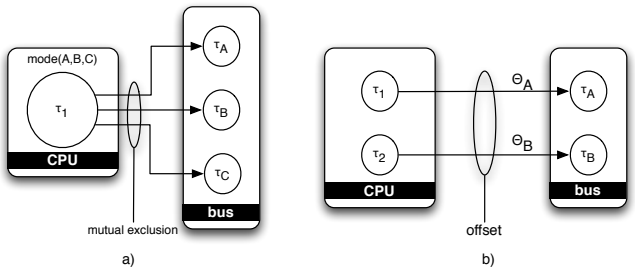

Fig. 3. Two possible dependencies in a system

Note that $\nu(\overrightarrow{\Theta}_{\overline{\Theta}}, \Delta\beta(n))$ and $\Delta\beta(n)$ are abstract functions which have to be concretized for the different types of dependencies. Next we will show how it is possible to describe two completely different kinds of dependencies with this new technique.

4.1 Mutual Exclusion

The first dependency we will consider is the mutual exclusion of event streams. Assume a task is transmitting different messages over a bus as depicted in figure 3 a). The transmission mode of the task depends on its input data. Only one message type can be sent in a transmission mode. In this case the message types exclude each other. Such a behavior can be described by the following lemma.

Lemma 2. *Let $\overrightarrow{\Theta}_{\overline{\Theta}}$ be a set of event streams having a mutual exclusion relation to each other. The limiting interval function is given by:*

$$\Delta\beta(n) = \min_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\Delta t(n, \Theta)\} \tag{4}$$

Proof. Mutual exclusion of event streams means that only one event stream $\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}$ is active at the time. Since the activation of these event streams is switching arbitrarily in a time interval it is necessary to determine for each time interval Δt which event stream delivers the most events in this interval. Therefore the maximum number of events in an interval is bounded by $\eta(\Delta t, \Theta^{+}_{\overline{\Theta}}) = \max_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\eta(\Delta t, \Theta)\}$. This can be transformed using equation (2) as follows:

$$\begin{aligned}
\eta(\Delta t, \Theta^{+}_{\overline{\Theta}}) &= \max_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\eta(\Delta t, \Theta)\} \\
\Delta\beta(n) &= \inf_{\Delta t \geq 0} \{\Delta t | \max_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\eta(\Delta t, \Theta)\} \geq n\} \\
\Delta\beta(n) &= \min_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{ \inf_{\Delta t \geq 0} \{\Delta t | \eta(\Delta t, \Theta) \geq n\}\} = \min_{\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\Delta t(n, \Theta)\}
\end{aligned}$$

Therefore the assumption holds. □

After defining the limiting interval function we will formulate how the concrete limiting event stream can be derived. Note that mode changes of tasks result in different execution times and have influence on the outgoing event streams. Therefore the outgoing event streams of one task are not necessarily identical in their aperiodic behavior. Only the periodic behavior is the same for all.

Lemma 3. *Via lemma 2 the limiting event stream for mutual exclusion for event streams* $\overrightarrow{\Theta}_{\overline{\Theta}}$ *having the same period and equal parameters concerning* N^{∞} *and* N^{P} *can be derived as follows:*

$$\nu(\overrightarrow{\Theta}_{\overline{\Theta}}, \Delta\beta(n)) = \bigcup_{i=1}^{N_{\Theta}^{\infty}} (\infty, \Delta\beta(i)) \cup \bigcup_{i=N_{\Theta}^{\infty}+1}^{N_{\Theta}^{\infty}+N_{\Theta}^{P}} (p, \Delta\beta(i)) \tag{5}$$

where $\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}$.

Proof. Let the minimum of two event tuples be defined as:

$$\min(\theta_1, \theta_2) := (\min(p_{\theta_1}, p_{\theta_2}), \min(a_{\theta_1}, a_{\theta_2}))$$

Furthermore let $\theta_{i,j}$ be the i-th event tuple of the j-th event stream and $\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}$. According to [8] each event stream $\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}$ can be transformed so that $\forall \Theta_i, \Theta_j \in \overrightarrow{\Theta}_{\overline{\Theta}}$: $|\Theta_i| = |\Theta_j|$ holds. From this it follows that

$$\bigcup_{i=1}^{|\Theta|} \min_{\Theta_j \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\theta_{i,j}\}$$

is a general upper bound for mutual exclusion.

As we consider only event streams produced by the same task, all outgoing event streams have the same periodic behavior. Therefore in conjunction with [8] all event streams in $\overrightarrow{\Theta}_{\overline{\Theta}}$ can be transformed so that N_{Θ}^{p} and N_{Θ}^{∞} are equal for all $\Theta \in \overrightarrow{\Theta}_{\overline{\Theta}}$. From this we can follow:

$$\begin{aligned}
\bigcup_{i=1}^{|\Theta|} \min_{\Theta_j \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\theta_{i,j}\} &= \bigcup_{i=1}^{N_{\Theta}^{\infty}} \min_{\Theta_j \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\theta_{i,j}\} \cup \bigcup_{i=N_{\Theta}^{\infty}+1}^{N_{\Theta}^{\infty}+P_{\Theta}^{\infty}} \min_{\Theta_j \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\theta_{i,j}\} \\
&= \bigcup_{i=1}^{N_{\Theta}^{\infty}} (\infty, \Delta\beta(i)) \cup \bigcup_{i=N_{\Theta}^{\infty}+1}^{N_{\Theta}^{\infty}+P_{\Theta}^{\infty}} (p, \Delta\beta(i))
\end{aligned}$$

Therefore the assumption holds. □

With the lemma stated above it is possible to consider mutual exclusion in distributed systems.

4.2 Offsets

In order to show the generality of our new approach we show how static offsets between tasks can be described by limiting event streams. For simplicity we consider here only strictly periodic stimulations, which are used in our case study. The methodology can also be used for arbitrary stimulations.

See figure 3 part b) where two different tasks are depicted each sending a message on a bus. All sending tasks have an offset to each other in order to prevent a peak load on the bus. This kind of modeling is often used for CAN bus communications. First of all we give a definition of the offsets.

Definition 7. *When a set of event streams $\overrightarrow{\Theta}_{\overline{\Theta}}$ has an offset based relation to each other, each event stream has an offset attribute ϕ_{Θ^+} describing an offset to a specific point in time.*

Next we define how the limiting interval function can be derived.

Lemma 4. *Let $\overrightarrow{\Theta}_{\overline{\Theta}}$ be a set of event streams having an offset relation to each other. And let $\tilde{\Theta}^+_\cup = \bigcup_{\Theta\in\overrightarrow{\Theta}_{\overline{\Theta}}} \bigcup_{\theta\in\Theta}(p_\theta, a_\theta + \phi_\Theta)$ be the normalized union of the offset-shifted event streams. Then the limiting interval function is given by:*

$$\Delta\beta(n) = \min_{j=1,\ldots,|\tilde{\Theta}^+_\cup|} (\Delta t(j+(n-1),\tilde{\Theta}^+_\cup) - \Delta t(j,\tilde{\Theta}^+_\cup)) \tag{6}$$

Proof. Since only static offsets are considered, the union of the offset shifted event streams describes the cumulated occurrence of the events. From this union we have to extract the minimum interval for each number of events. So we have to search over all possible combinations, which can be done by a sliding window approach. From this consideration it follows:

$$\Delta\beta(n) = \min_{j\in\mathbb{N}}(\Delta t(j+(n-1),\tilde{\Theta}^+_\cup) - \Delta t(j,\tilde{\Theta}^+_\cup))$$

So we have to show that the bound $j = 1,\ldots,|\tilde{\Theta}^+_\cup|$ in lemma 4 includes the minimal interval for a number of events. The following condition holds:

$$\Delta t(q,\tilde{\Theta}^+_\cup) - \Delta t(q - N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup) = p_{\tilde{\Theta}^+_\cup} : q > |\tilde{\Theta}^+_\cup|$$

So it follows for $j > |\tilde{\Theta}^+_\cup|$ that $\Delta t(j+n-1,\tilde{\Theta}^+_\cup) - \Delta t(j+n-1-N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup) = p_{\tilde{\Theta}^+_\cup}$ and $\Delta t(j,\tilde{\Theta}^+_\cup) - \Delta t(j - N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup) = p_{\tilde{\Theta}^+_\cup}$ holds. From this we can follow:

$$\begin{aligned}
&\Delta t(j+n-1,\tilde{\Theta}^+_\cup) - \Delta t(j,\tilde{\Theta}^+_\cup) \\
&= \Delta t(j+n-1,\tilde{\Theta}^+_\cup) - \Delta t(j,\tilde{\Theta}^+_\cup) - p_{\tilde{\Theta}^+_\cup} + p_{\tilde{\Theta}^+_\cup} \\
&= (\Delta t(j+n-1,\tilde{\Theta}^+_\cup) - p_{\tilde{\Theta}^+_\cup}) - (\Delta t(j,\tilde{\Theta}^+_\cup) - p_{\tilde{\Theta}^+_\cup}) \\
&= (\Delta t(j+n-1,\tilde{\Theta}^+_\cup) - \Delta t(j+n-1,\tilde{\Theta}^+_\cup) + \Delta t(j+n-1-N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup)) - \\
&\quad (\Delta t(j,\tilde{\Theta}^+_\cup) - \Delta t(j,\tilde{\Theta}^+_\cup) + \Delta t(j - N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup)) \\
&= (\Delta t(j+n-1-N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup)) - \Delta t(j - N^p_{\tilde{\Theta}^+_\cup},\tilde{\Theta}^+_\cup)
\end{aligned}$$

So we have shown that a minimum interval must also occur one period earlier and therefore also in the bound of the search interval. □

After having shown how the limiting interval function can be defined we will now introduce how the concrete event stream can be derived.

Lemma 5. *With the help of the last lemma we can derive the concrete limiting event stream for an offset-based dependency with $j = N^{\infty}_{\tilde{\Theta}^+_{\cup}}$ by the following equation:*

$$\nu(\overrightarrow{\Theta}_{\overline{\Theta}}, \Delta\beta(n)) = \bigcup_{i=1}^{j} (\infty, \Delta\beta(i)) \cup \bigcup_{i=j+1}^{j+N^{p}_{\tilde{\Theta}^+_{\cup}}} \left(p_{\tilde{\Theta}^+_{\cup}}, \Delta\beta(i) \right) \tag{7}$$

Proof. We have to show that the periodic behavior starts at the assumed value. Assume the following variables: $1 \leq j \leq |\tilde{\Theta}^+_{\cup}|$, $e = N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m + kN^{p}_{\tilde{\Theta}^+_{\cup}}$ and $1 \leq m \leq N^{p}_{\tilde{\Theta}^+_{\cup}}$ and the following equation giving the minimum interval for $N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m$ events where j is chosen accordingly:

$$\Delta\beta(N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m) = \Delta t(j + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j, \tilde{\Theta}^+_{\cup})$$

So we have to show that the following assumption holds for e events :

$$\Delta\beta(e) = \Delta\beta(N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m) + kp_{\tilde{\Theta}^+_{\cup}} \tag{8}$$

Assume $1 \leq j_1 \leq |\tilde{\Theta}^+_{\cup}|$ and $\Delta t(j_1 + (e-1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1, \tilde{\Theta}^+_{\cup})$ is the smallest interval for e events then it follows:

$$\begin{aligned}
&\Delta t(j_1 + (e-1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1, \tilde{\Theta}^+_{\cup}) \\
&= \Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m + kN^{p}_{\tilde{\Theta}^+_{\cup}} - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1, \tilde{\Theta}^+_{\cup}) \\
&= \Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m + kN^{p}_{\tilde{\Theta}^+_{\cup}} - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1, \tilde{\Theta}^+_{\cup}) + \\
&\quad \Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1), \tilde{\Theta}^+_{\cup}) \\
&= \Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j_1, \tilde{\Theta}^+_{\cup}) + kp_{\tilde{\Theta}^+_{\cup}}
\end{aligned}$$

But this cannot be smaller than assumption (8) because $kp_{\tilde{\Theta}^+_{\cup}}$ can not be smaller than $kp_{\tilde{\Theta}^+_{\cup}}$ and $\Delta t(j_1 + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1) - \Delta t(j_1, \tilde{\Theta}^+_{\cup})$ cannot be smaller than $\Delta t(j + (N^{\infty}_{\tilde{\Theta}^+_{\cup}} + m - 1), \tilde{\Theta}^+_{\cup}) - \Delta t(j, \tilde{\Theta}^+_{\cup})$. So the assumption holds. □

After having derived how static offsets can be described by limiting event streams we will now describe the worst-case response time analysis with the limiting event streams.

4.3 Worst-Case Response Time Analysis with Limiting Event Streams

In order to use limiting event streams it is necessary to adapt this new concept to the worst-case response time analysis from section 3.2. For this we have to determine the worst-case interference of tasks in an interval Δt when limiting event streams are considered. Due to space limitation we show only the sketch of the proof.

Lemma 6. *Let τ be the task under analysis and $\overline{\Theta}$ be a limiting event stream and $\Gamma_{\overline{\Theta},\tau,\tau_i} := \{\tau_j \in \Gamma_{hp,\tau} | (c^+_{\tau_j} > c^+_{\tau_i} \vee (c^+_{\tau_j} = c^+_{\tau_i} \wedge \rho_{\tau_j} > \rho_{\tau_i})) \wedge \Theta^+_{\tau_j} \in \overrightarrow{\Theta}_{\overline{\Theta}}\}$, then the worst-case response time is bounded by:*

$$r^+(\tau) = \max_{k\in[1,\ldots,m]} \{r^+(k,\tau) - \Delta t(k,\Theta^+_\tau)\} \quad m = \min_{k\in\mathbb{N}} \{k | r^+(k,\tau) \leq \Delta t(k+1,\Theta^+_\tau)\} \tag{9}$$

$$r^+(k,\tau) = \min\{\Delta t | \Delta t = b_\tau + k\cdot c^+_\tau + \sum_{\tau_i\in\Gamma_{hp,\tau}} \overline{\eta}(\Delta t,\tau_i,k,\tau)\cdot c^+_{\tau_i}\} \tag{10}$$

$$\overline{\eta}(\Delta t,\tau_i,k,\tau) = \min(\max(\min_{\forall\overline{\Theta}|\Theta^+_{\tau_i}\in\overrightarrow{\Theta}_{\overline{\Theta}}} \{\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta})\},0), \eta(\Delta t,\Theta^+_{\tau_i})) \tag{11}$$

$$\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta}) = \begin{cases} \eta(\Delta t,\Theta^+_{\overline{\Theta}}) - \sum_{\tau_j\in\Gamma_{\overline{\Theta},\tau,\tau_i}} \overline{\eta}(\Delta t,\tau_j,k,\tau) & \Theta^+_\tau \notin \overrightarrow{\Theta}_{\overline{\Theta}} \\ \eta(\Delta t,\Theta^+_{\overline{\Theta}}) - \sum_{\tau_j\in\Gamma_{\overline{\Theta},\tau,\tau_i}} \overline{\eta}(\Delta t,\tau_j,k,\tau) - k & \Theta^+_\tau \in \overrightarrow{\Theta}_{\overline{\Theta}} \end{cases} \tag{12}$$

Proof. Compared to (3), (10) differs only in the interference of the higher priority tasks. Equation (11) describes the number of task activations in a given interval Δt of higher priority tasks. It is the minimum of the amount described by the limiting event streams $\max(\min_{\forall\overline{\Theta}|\Theta^+_{\tau_i}\in\overrightarrow{\Theta}_{\overline{\Theta}}} \{\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta})\},0)$ and the amount described by the regular event stream of the task $\eta(\Delta t,\Theta^+_{\tau_i})$. If none of the event streams of the higher priority tasks is part of a limiting event stream, (11) is reduced to the regular event stream

$$\min(\max(\min_{\forall\overline{\Theta}|\Theta^+_{\tau_i}\in\overrightarrow{\Theta}_{\overline{\Theta}}} \{\},0), \eta(\Delta t,\Theta^+_{\tau_i})) = \min(\max(\infty,0), \eta(\Delta t,\Theta^+_{\tau_i})) = \eta(\Delta t,\Theta^+_{\tau_i}))$$

resulting in (3) and (10) being identical.

So we have to show that the interference is maximal if limiting event streams are considered. Let $\sum_{\tau_i\in\Gamma_{hp,\tau}} \overline{\eta}(\Delta t,\tau_i,k,\tau)\cdot c^+_{\tau_i}$ be the maximum load of higher priority tasks in an interval Δt. Since every $\overline{\Theta} : \Theta^+_{\tau_i} \in \overrightarrow{\Theta}_{\overline{\Theta}}$ is a valid bound for the number of events in an interval of a task τ_i, and $\eta(\Delta t,\Theta^+_{\tau_i})$ is the regular bound for the maximum stimulation of a task in an interval Δt, we can follow:

$$\overline{\eta}(\Delta t,\tau_i,k,\tau) = \min(\max(\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta}),0), \eta(\Delta t,\Theta^+_{\tau_i})) \tag{13}$$

where $\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta})$ is the maximum number of events limited by the liminting event stream $\overline{\Theta}$ for task τ_i. Since for one limiting event stream the condition $\eta(\Delta t,\Theta^+_{\overline{\Theta}}) \leq \sum_{\Theta\in\overrightarrow{\Theta}_{\overline{\Theta}}} \eta(\Delta t,\Theta)$ holds we can follow $\eta(\Delta t,\Theta^+_{\overline{\Theta}}) = \sum_{\tau_i|\Theta^+_{\tau_i}\in\overrightarrow{\Theta}_{\overline{\Theta}}} \overline{\eta}(\Delta t,\tau_i,k,\tau)$. From this it follows: $\overline{\overline{\eta}}(\Delta t,\tau_i,k,\tau,\overline{\Theta}) = \eta(\Delta t,\Theta^+_{\overline{\Theta}}) - \sum_{\tau_j|\Theta^+_{\tau_j}\in\overrightarrow{\Theta}_{\overline{\Theta}}/\Theta^+_{\tau_i}} \overline{\eta}(\Delta t,\tau_j,k,\tau)$. This can

be inserted in (13) and we get:

$$\overline{\eta}(\Delta t,\tau_i,k,\tau)=\min(\max(\eta(\Delta t,\Theta^+_{\overline{\Theta}})-\sum_{\tau_j|\Theta^+_{\tau_j}\in\overrightarrow{\Theta}_{\overline{\Theta}}/\Theta^+_{\tau_i}}\overline{\eta}(\Delta t,\tau_j,k,\tau),0),\eta(\Delta t,\Theta^+_{\tau_i}))$$

In order to achieve that the interference of the higher priority tasks $\Gamma_{hp,\tau}$ is maximal, the events must be distributed so that the task with the greatest worst-case execution time is triggered as often as possible then the task with the second greatest worst-case execution time and so on. For tasks having the same worst-case execution time, the order for the distribution is given by the priority. From this it follows that the task τ_i is subject to the maximum interference when the available events $\eta(\Delta t,\Theta^+_{\overline{\Theta}})$ are first distributed on the tasks in the taskset $\Gamma_{\overline{\Theta},\tau,\tau_i}:=\{\tau_j\in\Gamma_{hp,\tau}|(c^+_{\tau_j}>c^+_{\tau_i}\vee(c^+_{\tau_j}=c^+_{\tau_i}\wedge\rho_{\tau_j}>\rho_{\tau_i}))\wedge\Theta^+_{\tau_j}\in\overrightarrow{\Theta}_{\overline{\Theta}}\}$ and the remaining events on task τ_i:

$$\overline{\eta}(\Delta t,\tau_i,k,\tau)=\min(\max(\eta(\Delta t,\Theta^+_{\overline{\Theta}})-\sum_{\tau_j\in\Gamma_{\overline{\Theta},\tau,\tau_i}}\overline{\eta}(\Delta t,\tau_j,k,\tau),0),\eta(\Delta t,\Theta^+_{\tau_i}))$$

Since all limting event streams are a bound for the number of events we have to take the overall minimum and we finally get:

$$\overline{\eta}(\Delta t,\tau_i,k,\tau)=\min(\max(\min_{\forall\overline{\Theta}|\Theta^+_{\tau_i}\in\overrightarrow{\Theta}_{\overline{\Theta}}}\{\eta(\Delta t,\Theta^+_{\overline{\Theta}})-\sum_{\tau_j\in\Gamma_{\overline{\Theta},\tau,\tau_i}}\overline{\eta}(\Delta t,\tau_j,k,\tau)\},0),\eta(\Delta t,\Theta^+_{\tau_i}))$$

Therefore the assumption holds. The proof of equation (12) case two includes only k jobs, because the task τ under analysis is part of the limitig event stream. The proof for this case is analog to the last one. □

As initial value for the fix-point iteration in equation (10) we take $\Delta t(k,\Theta^+_\tau)+c^+_\tau$. Note that the complexity of the response time analysis is still pseudo-polynominial. The complexity to calculate the limiting event streams depends on the kind of the dependency which is considered. The analysis, however, is not affected by this problem. So it is reasonable to find upper bounds for the limiting event streams to improve the runtime performance.

5 Example and Results

In order to show the impact of the introduced new technique we consider a 500 kBit/s CAN bus. This automotive application is depicted in figure 4. The system comprises

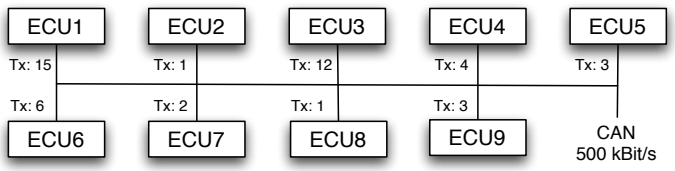

Fig. 4. Can bus communication of an automotive architecture

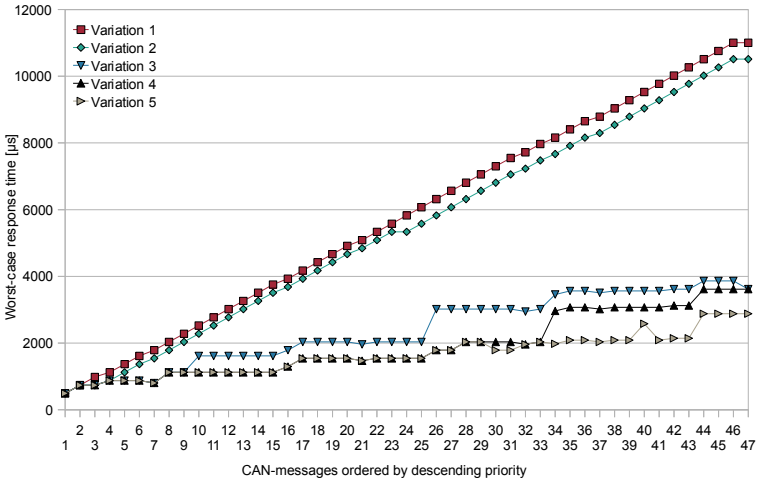

Fig. 5. Absolute worst-case response times of the messages

nine electronic control units transmitting data on the CAN bus. The number of different messages sent by each ECU is stated as "Tx" in the illustration. There are 47 messages in total, which all are transmitted periodically. All stimulations have an initial offset to each other in order to prevent peak load on the bus and to provide smaller response times of the messages. These offsets are defined by the designer of the system. Furthermore two pairs of messages (one pair on ECU3 and one pair on ECU6) have a mutual exclusion dependency.

The system has been analyzed in five different levels of detail. In the *first variation* we have analyzed the system without considering any dependencies. As *second variation* only the mutual exclusion of the messages has been considered. The *third variation* additionally includes the offsets with only one limiting event stream over all message stimulations from an ECU as depicted in figure 2 part a). The *fourth variation* increases the level of detail of the offset consideration by the variation represented in figure 2 part b). The last and therefore the *fifth variation* increases also the level of detail of the offset dependencies by the variation c) of figure 2. The results are depicted in figure 5.

As expected, including more dependencies and therefore increasing the level of detail leads to better worst-case response times of the tasks. The consideration of the mutual exclusion leads to a nearly constant improvement of the response times of all messages, which can be seen in figure 5. This does not apply to the consideration of the offset dependencies. As can be seen for variation 3, the impact on the messages

Variation 1	Variation 2	Variation 3	Variation 4	Variation 5
10020 μs	9528 μs	3618 μs	3126 μs	2142 μs

Table 1. Worst-Case response times of message 42

Variation 1	Variation 2	Variation 3	Variation 4	Variation 5
82 ms	92 ms	301 ms	821 ms	1428 ms

Table 2. Runtimes of the different variations

is different. The next level of detail leads to an improvement of the response times of messages 10 to 47. The last curve describes the highest level of detail and leads to a further improvement of the response times for messages with a small priority. So it can be seen that the different possibilities to integrate the dependencies into the analysis lead to very different results. For a better impression of the improvements we have described the worst-case response times of message 42 in table 1.

Based on the discussion above we now consider the corresponding runtimes shown in the table 2. All runs were performed on a machine with a 2.66 GHz Intel processor and 4 GB of system memory. The runtime when considering no dependencies is the shortest. The inclusion of mutual exclusion leads to almost no increase of the runtime. The inclusion of offsets results in a noteworthy increase of the runtime. The simple case of consideration of the offset dependency leads to nearly 209 ms of more runtime. The next improvement gains an additional 520 ms of runtime with the benefit that many messages have better worst-case response times. The last variation and therefore the most detailed has a runtime of 1428 ms, but only some messages have a better response time in this variations.

Variation 3 is best suited for a design space exploration because here the quotient of the runtime increase and the average reduction of response times is minimal. As mentioned in the introduction, another possibility is to start the design process with nearly no dependency correlation and to increase the level of detail in later design iterations where more accurate results are necessary.

6 Conclusion

In this paper we have introduced a holistic model to describe task dependencies in distributed real-time systems. The new approach has been applied to a real automotive architecture. We have shown for two different dependencies how they can be described by this new model. We have cut the complexity of the dependencies from the real-time analysis, which has not been achieved in previous work. Additionally we have shown how the consideration of dependencies can be approximated very easily by this new concept.

Finally a case study has been conducted to show the improvements of the approach. It has been shown that our model works for different kinds of dependencies. Furthermore we have shown that the introduced new approximation technique can lead to significant improvements of the runtimes. In the future we will show how to integrate other dependencies into this new model.

References

1. Albers, K., Slomka, F.: An event stream driven approximation for the analysis of real- time systems. In: ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems. pp. 187–195. IEEE (July 2004)
2. Gresser, K.: An event model for deadline verification of hard real-time systems. In: Proceedings of the 5th Euromicro Workshop on Real-Time Systems (1993)
3. Gutierrez, J.C.P., Harbour, M.G.: Schedulability analysis for tasks with static and dynamic offsets. In: RTSS. p. 26 ff (1998)
4. Gutierrez, J.C.P., Harbour, M.G.: Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In: IEEE Real-Time Systems Symposium. pp. 328–339 (1999)
5. Henia, R., Ernst, R.: Context-aware scheduling analysis of distributed systems with tree-shaped task-dependencies. In: DATE '05: Proceedings of the conference on Design, Automation and Test in Europe. pp. 480–485 (2005)
6. Henia, R., Ernst, R.: Improved offset-analysis using multiple timing-references. In: DATE '06: Proceedings of the conference on Design, automation and test in Europe. pp. 450–455 (2006)
7. Jonas Rox, R.E.: Exploiting inter-event stream correlations between output event streams of non-preemptively scheduled tasks. In: Proc. Design, Automation and Test in Europe (DATE 2010) (March 2010)
8. Kollmann, S., Albers, K., Slomka, F.: Effects of simultaneous stimulation on the event stream densities of fixed-priority systems. In: Spects'08: Proceedings of the International Simulation Multi-Conference. IEEE (June 2008)
9. Kollmann, S., Pollex, V., Slomka, F.: Holisitc real-time analysis with an expressive event model. In: proceedings of the 13th Workshop of Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (2010)
10. Kuenzli, S., Hamann, A., Ernst, R., Thiele, L.: Combined approach to system level performance analysis of embedded systems. In: CODES+ISSS '07: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis. pp. 63–68. ACM, New York, NY, USA (2007)
11. Lehoczky, J.P.: Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proceedings of the 11th IEEE Real-Time Systems Symposium. pp. 201–209 (December 1990)
12. Pellizzoni, R., Lipari, G.: Improved schedulability analysis of real-time transactions with earliest deadline scheduling. In: RTAS '05: Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium. pp. 66–75 (2005)
13. Redell, O.: Analysis of tree-shaped transactions in distributed real-time systems. In: ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS'04). pp. 239–248 (2004)
14. Richter, K.: Compositional Scheduling Analysis Using Standard Event Models - The SymTA/S Approach. Ph.D. thesis, University of Braunschweig (2005)
15. Tindell, K.: Adding time-offsets to schedulability analysis. Tech. rep., University of York, Computer Science Dept, YCS-94-221 (1994)
16. Tindell, K., Clark, J.: Holistic schedulability analysis for distributed hard real-time systems. Microprocessing and Microprogramming 40, 117–134 (April 1994)
17. Wandeler, E.: Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems. Ph.D. thesis, ETH Zurich (September 2006)