# Reducing Response Times by Competition Based Dependencies

Steffen Kollmann, Victor Pollex, and Frank Slomka
Ulm University
firstname.lastname@uni-ulm.de

**Abstract**

To ensure that time constraints in real-time systems are satisfied it is necessary to verify the real-time behavior during the design process of such systems. Schedulabilitiy analysis approaches can be used for this. The disadvantage of these methods is that sometimes the calculated bounds are an overestimation of the real behavior of the system. Therefore it is necessary to include the system contexts into the analysis to achieve tight bounds. In this paper we will introduce a system context that is based on the competition of tasks for a common resource. This dependency exists inherently in every system.

## 1. Introduction

Designing embedded systems is challenging due to the steadily increasing requirements and architectural complexity. A good example for this trend can be seen in the automotive industry where premium class cars have over 100 electronic components comprising electronic control units (ECUs), buses, etc. During the design process of such systems the functional and non-functional requirements have to be fulfilled.
One non-functional requirement which has to be verified during an early design phase are the real-time constraints, for example those of the various safety critical functions of an automotive. Bounds for the worst-case behavior of the system are determined by performing a schedulability analysis and are then compared to the real-time constraints. The more contexts of the system are considered, the better the bounds determined by the analysis methods become. Disregarding system contexts in the analysis can cause an overestimation of the bounds leading to overdimensioned systems and thus increase the costs of the systems unnecessarily. In previous work different types of dependencies have been considered, like mutual exclusion of tasks and offsets between the stimulation of the tasks or task chains, but for many other dependencies it is still open how to consider them in the analysis.
This paper presents a new kind of dependency which exists inherently in each distributed system design. In order to include this into the real-time analysis we use an existing method proposed in [KPKS10], where a general model for the inclusion of different kinds of dependencies into the analysis is proposed.
The remainder of the paper is organized as follows: In section 2 an overview of the related work is given. The system model is defined in section 3. The resulting real-time analysis is presented in section 4. An example shows the impact of the newly introduced dependency on the response times in a distributed system. Finally a conclusion is given.

## 2. Related Work

Most of the considered related work uses the periodic event model with jitter. Tindel and Clark have introduced in [TC94] the real-time analysis for distributed real-time systems. The lack of this approach is that the task stimulations are considered as independent and therefore for each task the worst-case response time is calculated separately. The transaction model proposed in [Tin94] copes with this problem and allows the description of static offsets between tasks. To further improve this Gutierrez et al. [GH98] extended the model to dynamic offsets so that the offset can vary from one job of a task to another. Additionally, they have introduced an idea about mutual exclusion of tasks [GH99] which is based on offsets between tasks. Redell has enhanced the idea to tree-shaped dependencies [Red04], since Gutierrez et al. have only considered simple task chains. Pellizoni et al. have shown that this model is also applicable to dynamic schedules and applied the transaction model to earliest deadline first scheduling in [PL05].
Henia et al. have used the SymTA/S approach [Ric05] to extend the idea of the transaction model in order to introduce timing-correlations between tasks in parallel paths in distributed systems [HE05]. This idea has then been improved in [HE06].
The description of dependencies is not bounded to the transaction model. Many other correlations have been described in related work. So, for example, another dependency considering the simultaneous occurrence of events is presented by Kollmann et al. in [KAS08] and recently, Rox et al. [RE10] have described a correlation between tasks which is caused by a non-preemptive scheduler. However the dependency we will consider in this paper is only subject to the available capacity of the resource and is valid regardless of the scheduling policy used.

## 3. System Model

As we use the approach presented in [KPKS10] to describe our new dependency we will repeat here the important points of that system model. We divide our model into a task model and an event model.

### 3.1. Task Model

We abstractly consider the applications in a distributed system as tasks. A task can be a process on a processing unit or a message transmission on a bus. The tasks ($\tau_i$) mapped on a resource are grouped into a taskset $\Gamma = \{\tau_1, ..., \tau_n\}$. A task is defined as follows:

**Definition 1.** *A task is a tuple* $\tau = (c^+, c^-, d, \phi, \Theta^+, \dot{\Theta}^+)$ *consisting of* $c^+$ *the worst-case execution time,* $c^-$ *the best-case execution time, d the relative deadline of the task,* $\phi$ *the priority of the task (lower value = higher priority),* $\Theta^+$ *the maximum incoming stimulation and* $\dot{\Theta}^+$ *the maximum outgoing stimulation.*

Let $\tau_{i,j}$ be the j-th job/execution of task $\tau_i$. We assume that each job of a task generates an event at the end of its execution to notify other tasks. Further we define that $\Gamma_{HP,\tau}$ is a taskset containing all tasks mapped on the same resource and having a higher priority than task $\tau$.

### 3.2. Event Model

As event model we use the event streams proposed by Gresser [Gre93]. The basic idea is to define an event function $\eta(\Delta t, \Theta^+)$ which can calculate the maximum number of events occurring within an interval of length $\Delta t$ (when speaking of an interval, we mean the length of the interval).

The idea of the 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 behaviour. This is because the minimum interval for $n$ events cannot be smaller than the minimum interval for $n-1$ events since the first interval also includes $n-1$ events.

**Definition 2.** *A maximum event stream $\Theta^+ = \{\theta_1, \theta_2, ..., \theta_n\}$ is a set of event stream elements $\theta$ and each event stream element $\theta = (p, a)$ consists of a period $p$ and an offset-interval $a$. The maximum event stream satisfies 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 $\theta$ describes a set of intervals $\{a_\theta + k \cdot p_\theta | k \in \mathbb{N}\}$ of the sequence. With an infinite ($\infty$) period it is possible to model irregular behaviour. 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 3.** *The event function denotes for an event stream $\Theta^+$ and an interval $\Delta t$ the corresponding number of events:*

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

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

**Definition 4.** *The interval function denotes for an event stream $\Theta^+$ and a number of n events the corresponding minimum interval in which these events can occur:*

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

A detailed definition of the concept and the mathematical foundation of the event streams can be found in [AS04]. Event streams can be described in several ways by the event stream model. For an efficient implementation of the approach we normalize the event streams.

**Definition 5.** *A normalized event stream $\tilde{\Theta}^+$ has the form:*

$$\{(\infty, a_1), \ldots, (\infty, a_m), (p, a_{m+1}), \ldots, (p, a_n)\} : (1 \leq m \leq n \wedge a_i \leq a_j \Leftrightarrow i \leq j \wedge a_n - a_{m+1} \leq p) \tag{3}$$

Meaning that the event stream has first an aperiodic part and then a periodic part where each periodic element has the same period. Furthermore all elements are sorted by their offsets. We define also 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 we can formulate the utilization of a task $\tau$ as follows:

**Definition 6.** *The utilization of a task $\tau$ is defined as:*

$$U_\tau = \frac{N^{p}_{\tilde{\Theta}^+_\tau} \cdot c^+_\tau}{p_{\tilde{\Theta}^+_\tau}} \tag{4}$$

We define further that event streams which are only monotonically increasing are denoted by $\Theta$. The previous definitions apply also for these event streams.

## 4. Holistic Real-Time Analysis

Due to space limitation we will give here only a short summary of the holistic real-time analysis. In [TC94] it is described that in each global iteration step (this is the iteration over all tasks in the system) of the real-time analysis the worst-case response time and the outgoing stimulation for each task in the system are computed until a fixed point is found. This approach with some extension has been used in [KPS10] to perform a holistic real-time analysis with event streams. Since the paper at hand focuses on the improvement of the worst-case response time of a task we will only repeat this part of the analysis here. Based on Lehoczky's et al. [Leh90] worst-case response time analysis we can define the analysis for event streams as follows:

**Definition 7.** *The worst-case response time of a task is bounded by:*

$$\begin{aligned} r^+(\tau) &= \max_{k\in[1,\ldots,m]}\{r^+(k,\tau)-\Delta t(k,\Theta_\tau^+)\}\ m=\min_{k\in\mathbb{N}}\{k|r^+(k,\tau)\leq\Delta t(k+1,\Theta_\tau^+)\} \\ r^+(k,\tau) &= \min\{\Delta t|\Delta t=b_\tau+k\cdot c_\tau^+ + \sum_{\tau_i\in\Gamma_{hp,\tau}}\eta(\Delta t,\Theta_{\tau_i}^+)\cdot c_{\tau_i}^+\} \end{aligned} \tag{5}$$

The proof and an explanation is given in [Leh90].

### 4.1. Limiting Event Stream

The lack of the previous discussed model is that the context of the system is not considered for the analysis. In [KPKS10] a general model to describe dependencies has been proposed. This model has been successfully applied to a real automotive architecture in [KPK+10]. The idea is to separate the worst-case response time analysis from the determination of the dependencies. We repeat here the main points and extensions of the model. On this basis we are able to derive our new dependency.

**Definition 8.** *The limiting event stream is an event stream which defines the maximum occurrence of events for a set of event streams. The limiting event stream is defined as* $\overline{\Theta}=(\Theta,\overrightarrow{\Theta})$. $\Theta$ *describes the limitation for a set of event streams* $\overrightarrow{\Theta}$. *The limiting event stream fulfills the condition:*

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

Note that for the limitation $\Theta_{\overline{\Theta}}$ we only demand that the resulting event function is monotonically increasing. It is not necessary that the bound is subadditive. It must only be a valid bound for a number of events.

**Example 1.** *Assume an interval of* $\Delta t=25t.u.$ *and two event streams* $\Theta_A^+=\Theta_B^+=\{(100,0)\}$ *and an offset dependency of* $50t.u.$ *between these two event streams. If the offset is not considered in the interval of* $25t.u.$ *two events occur. One from each event stream, because in this case the simple union of the two event streams describes the behavior* $\eta(\Delta t,\Theta_A^+\cup\Theta_B^+)=2$. *If the dependency is considered, the limiting event stream can be described as* $\overline{\Theta}=(\{(100,0),(100,50)\},\{\Theta_A^+,\Theta_B^+\})$. *In this case in the interval of* $25t.u.$ *only one event occurs, because* $\overline{\Theta}$ *bounds the cumulated number of the events* $\eta(\Delta t,\Theta_{\overline{\Theta}})=1$. *Therefore the number of preemptions in an interval can be reduced by the use of limiting event streams.*

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

**Definition 9.** *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 dependency from a given 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:*

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

Note that $\nu(\overrightarrow{\Theta}_{\overline{\Theta}}, \Delta\beta(n))$ and $\Delta\beta(n)$ are abstract formulas which must be concretely formulated for the different types of dependencies. Next we will show how the limiting event streams can be used to improve the worst-case response time analysis.

**Lemma 1.** *Let $\tau$ be the task under analysis, $\overline{\Theta}$ a limiting event stream, and $\Gamma_{\overline{\Theta},\tau,\tau_i} := \{\tau_j \in \Gamma_{hp,\tau} | (c^+_{\tau_j} > c^+_{\tau_i} \vee (c^+_{\tau_j} = c^+_{\tau_i} \wedge \phi_{\tau_j} > \phi_{\tau_i})) \wedge \Theta^+_{\tau_j} \in \overrightarrow{\Theta}_{\overline{\Theta}}\}$, then the worst-case response time is bounded by:*

$$r^+_{les}(\tau) = \max_{k \in [1,\ldots,m]} \{r^+(k,\tau) - \Delta t(k, \Theta^+_\tau)\} \quad m = \min_{k \in \mathbb{N}} \{k | r^+(k,\tau) \leq \Delta t(k+1, \Theta^+_\tau)\} \tag{6}$$

$$r^+(k,\tau) = \min\{\Delta t | \Delta t = b_\tau + k \cdot c^+_\tau + \sum_{\tau_i \in \Gamma_{hp,\tau}} \overline{\eta}(\Delta t, \tau_i, k, \tau) \cdot c^+_{\tau_i}\} \tag{7}$$

$$\overline{\eta}(\Delta t, \tau_i, k, \tau) = \min(\max(\min_{\forall \overline{\Theta} | \Theta^+_{\tau_i} \in \overrightarrow{\Theta}_{\overline{\Theta}}} \{\overline{\overline{\eta}}(\Delta t, \tau_i, k, \tau, \overline{\Theta})\}, 0), \eta(\Delta t, \Theta^+_{\tau_i})) \tag{8}$$

$$\overline{\overline{\eta}}(\Delta t, \tau_i, k, \tau, \overline{\Theta}) = \begin{cases} \eta(\Delta t, \Theta_{\overline{\Theta}}) - \sum_{\tau_j \in \Gamma_{\overline{\Theta},\tau,\tau_i}} \overline{\eta}(\Delta t, \tau_j, k, \tau) & \Theta^+_\tau \notin \overrightarrow{\Theta}_{\overline{\Theta}} \\ \eta(\Delta t, \Theta_{\overline{\Theta}}) - \sum_{\tau_j \in \Gamma_{\overline{\Theta},\tau,\tau_i}} \overline{\eta}(\Delta t, \tau_j, k, \tau) - k & \Theta^+_\tau \in \overrightarrow{\Theta}_{\overline{\Theta}} \end{cases} \tag{9}$$

*Proof.* The proof is given in [KPKS10]. □

The idea is to reduce the interference of the higher priority tasks. By means of the limiting event streams the number of preemptions of higher priority tasks is bounded. This reduced number of events for a set of maximum event streams must be distributed on the possible number of events of the maximum event streams for which the limitation holds. This distribution is not done arbitrarily. It has to be ensured that the interference in each iteration step is maximal. For this the events are distributed so that the task with the greatest worst-case execution time gets as many events as possible, then the task with the second greatest worst-case execution time and so on. Thereby, the worst-case response time analysis is not affected by the kind of dependency. Each dependency in the system is abstractly described by a limiting event stream. Therefore for each kind of dependency the concrete limitation has to be derived.
Note that we have discussed here a general worst-case response time bound for fixed priority systems which applies for non-preemptive and preemptive systems. Better bounds can be given for each scheduling policy which differ slightly from equation (5), as described in [GRS96]. For the example in section 6 we use the scheduling specific bounds, but in the theory part we have considered only a general bound. Since the dependencies are described orthogonal to the analysis, it is insignificant for which of the bounds the improvement is shown.

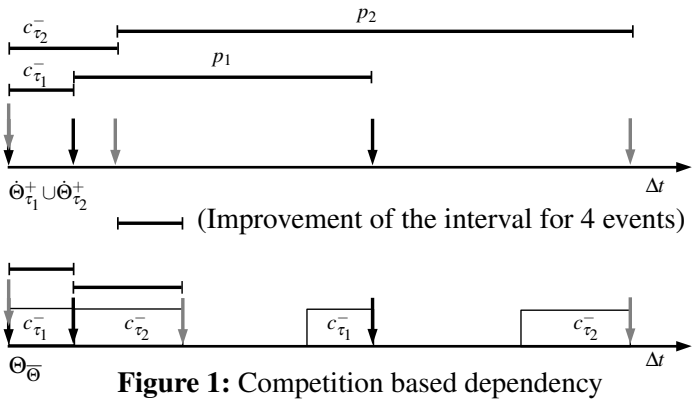



**Figure 1:** Competition based dependency

## 5. Competition Based Dependency

Based on the system model and the real-time analysis we introduce in this section a new kind of dependency called competition based dependency. Assume the tasks $\tau_1$ and $\tau_2$ are executed by the same resource, which means that they compete for the resource. In related work, for example [TC94], the outgoing event streams are considered independently for the analysis of the tasks $\tau_1$ or $\tau_2$, but this can lead to an overestimation of the real-time behavior of the system.
Let us consider the gantt-chart in figure 1 and that $\dot{\Theta}^+_{\tau_1} = \{(\infty,0),(p_1,c^-_{\tau_1})\}$ and $\dot{\Theta}^+_{\tau_2} = \{(\infty,0),(p_2,c^-_{\tau_2})\}$. In the upper part of the gantt-chart the case of non competing tasks is considered, meaning that the distance between the outgoing events are computed independently. In this case the maximum cumulated number of events produced by $\tau_1$ and $\tau_2$ is calculated by: $\eta(\Delta t, \dot{\Theta}^+_{\tau_1} \cup \dot{\Theta}^+_{\tau_2})$. But this is not always possible, since the jobs $c^-_{\tau_1}$ and $c^-_{\tau_2}$ must be executed sequentially, because $\tau_1$ and $\tau_2$ are executed by the same resource. This means that four events can only occur in a minimal distance of $\Delta t = c^-_{\tau_1} + c^-_{\tau_2}$ and not in the interval $\Delta t = c^-_{\tau_2}$. This is depicted in the lower part which describes the correct occurrence of the events. Due to the task interference it is not sufficient to consider the outgoing event streams independently from each other.
This interference can be modeled by a limiting event stream. Note, that this dependency is independent of the scheduling policy. Therefore this consideration of context is more general than the one proposed in the related work. So the simple question for this new context is: Has the resource enough capacity to produce the n events?
To calculate the limiting event stream we have to determine the minimal distance between $n$ events by formulating the limiting interval function for competition based dependencies.

**Lemma 2.** *Let $\Gamma_R$ be a subset of m tasks sharing the same processor and $N = \{(n_1,\ldots,n_m) : \sum_{i=1}^{m} n_i = n\}$ the set of distributions of n events, where each task $\tau_i \in \Gamma_R$ produces $n_i$ events, then the limiting interval function is given by:*

$$\Delta\beta(n) = \min_{(n_1,\ldots,n_m)\in N} \left( \max \left( \max_{i=1,\ldots,m} \left( \Delta t \left( n_i, \dot{\Theta}^+_{\tau_i} \right) \right), \left( \sum_{i=1}^{m} (n_i - 1) \cdot c^-_{\tau_i} \right) \right) \right) \tag{10}$$

*Proof.* Assume $\Theta^+_{\cup} = \bigcup_{\tau_i \in \Gamma_R} \dot{\Theta}^+_{\tau_i}$ as the case of independent stimuli and assume further that

$\Delta t(n,\Theta^+_\cup) = \inf\{\Delta t | \sum_{\tau_i \in \Gamma_R} \eta(\Delta t, \dot{\Theta}^+_{\tau_i}) \geq n\} = \Delta t_0 \wedge n_i = \eta(\Delta t_0, \dot{\Theta}^+_{\tau_i})$ then it follows:

$$\begin{aligned}
& \forall i \in [1,\ldots,m] : \Delta t(n_i, \dot{\Theta}^+_{\tau_i}) \leq \Delta t_0 \\
\Rightarrow \quad & \max_{i\in[1,\ldots,m]} \{\Delta t(n_i, \dot{\Theta}^+_{\tau_i})\} \leq \Delta t_0 \\
\Rightarrow \quad & \min_{\Sigma n_i = n} (\max\{\Delta t(n_i, \dot{\Theta}^+_{\tau_i})\}) \leq \Delta t_0 \\
\Rightarrow \quad & \min_{(n_1,\ldots,n_m)\in N} \left( \max_{i=1,\ldots,m} \left( \Delta t \left( n_i, \dot{\Theta}^+_{\tau_i} \right) \right) \right) \leq \Delta t_0
\end{aligned} \tag{11}$$

This bound describes the minimum interval in which $n$ events can occur considering the case of independence. This can be relaxed, because the jobs must be executed sequentially. So we derive a lower bound by the execution demand.
Between $n_i$ outgoing events from a task $\Delta t(n_i, \dot{\Theta}^+_{\tau_i})$ at least $(n_i - 1)c^-_{\tau_i}$ execution demand must be executed. So the minimum interval $\Delta t_0$ in which n events can be produced must be greater than:

$$\Delta t_0 \geq \sum_{i=1}^{m} (n_i - 1) c^-_{\tau_i} \tag{12}$$

We can combine (11) and (12) and get (13):

$$\min_{(n_1,\ldots,n_m)\in N} \left( \max \left( \max_{i=1,\ldots,m} \left( \Delta t \left( n_i, \dot{\Theta}^+_{\tau_i} \right) \right), \left( \sum_{i=1}^{m} (n_i - 1) \cdot c^-_{\tau_i} \right) \right) \right) \tag{13}$$

□

Next we show how the concrete event stream can be derived:

**Lemma 3.** *By means of lemma 2 the concrete limiting event stream can be derived. Let $\tilde{\Theta}^+_\cup$ be the normalized union of the set $\overrightarrow{\Theta}_{\overline{\Theta}} = \{\dot{\Theta}^+_{\tau_i}\}$ and $j = min\{i | i > N^\infty_{\tilde{\Theta}^+_\cup} \wedge \forall a \in [i, i + N^p_{\tilde{\Theta}^+_\cup}] : \Delta\beta(a) \leq \Delta t(a, \tilde{\Theta}^+_\cup))\}$. Then the concrete limiting event stream can be derived as follows:*

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

*Proof.* Assume an utilization less than one on the processor then it follows:

$$\sum_{\tau_i} \frac{N^p_{\dot{\Theta}^+_{\tau_i}} c^+_{\tau_i}}{p_{\dot{\Theta}^+_{\tau_i}}} < 1 \Rightarrow \frac{\sum_{\tau_i} N^p_{\dot{\Theta}^+_{\tau_i}} c^+_{\tau_i}}{p_{\tilde{\Theta}^+_\cup}} < 1 \Rightarrow \sum_{\tau_i} N^p_{\dot{\Theta}^+_{\tau_i}} c^+_{\tau_i} < p_{\tilde{\Theta}^+_\cup} \tag{14}$$

First we have to show that when an idle time exists, a periodic behavior can be assumed. From (14) together with $\Delta\beta(a) \leq \Delta t(a, \tilde{\Theta}^+_\cup) : a > N^\infty_{\tilde{\Theta}^+_\cup}$ it follows:

$$\Delta\beta(a) + \sum_{\tau_i} N^p_{\dot{\Theta}^+_{\tau_i}} c^+_{\tau_i} \leq \Delta t(a, \tilde{\Theta}^+_\cup)) + p_{\tilde{\Theta}^+_\cup} \Rightarrow \Delta\beta(a + N^p_{\Theta^+_\cup}) = \Delta t(a + N^p_{\Theta^+_\cup}, \tilde{\Theta}^+_\cup))$$

This means that after an idle time only equation (11) can hold for $a + k \cdot N^{p}_{\tilde{\Theta}^{+}_{\cup}}$ events. Now we have to show that for each periodic element such a gap must exist. From (14) we can follow that $\varepsilon$ idle time is saved in each period. So we can conclude for each periodic element of the normalized event stream $\tilde{\Theta}^{+}_{\cup}$:

$$x + kp = x + k\left(\sum_{\tau_i} N^{p}_{\dot{\Theta}^{+}_{\tau_i}} c^{+}_{\tau_i} + \varepsilon\right) \tag{15}$$

Assume that at a point in time $x$ an amount of execution demand is requested so that:

$$x < \sum_{\tau_i} \eta(x, \dot{\Theta}^{+}_{\tau_i}) c^{+}_{\tau_i}$$

Together with (15) this execution demand must be executed after

$$k = \left\lceil \frac{\sum_{\tau_i} \eta(x, \dot{\Theta}^{+}_{\tau_i}) c^{+}_{\tau_i} - x}{\varepsilon} \right\rceil$$

periods. From this together with (11) it follows:

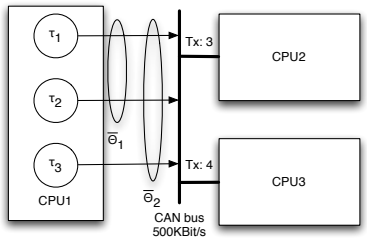
$$\Delta\beta(\eta(x, \tilde{\Theta}^{+}_{\cup}) + kN^{p}_{\tilde{\Theta}^{+}_{\cup}}) = \Delta t(\eta(x, \tilde{\Theta}^{+}_{\cup}) + kN^{p}_{\tilde{\Theta}^{+}_{\cup}})$$

Hence, for each periodic event tuple an idle time exists and therefore a periodic behaviour. □

By means of the last two lemma it is possible to calculate the competition based dependency in a distributed system.

## 6. Example

In order to show that our new kind of dependency has a direct impact on real applications we have analyzed a distributed system depicted in figure 2a. The system consists of three processor units connected by a fixed priority non-preemptive communication channel. This communication channel is a 500 kBit/s CAN bus with 10 messages. Note that the messages are modeled as tasks. Three messages are transmitted by CPU1, three messages by CPU2 and four messages by CPU3.




**(a)** Architecture

| Task | $c^{+}[\mu s]$ | $c^{-}[\mu s]$ | $d[\mu s]$ | $\phi$ | $\Theta^{+}$ |
|---|---|---|---|---|---|
| $\tau_1$ | 500 | 250 | $\infty$ | 1 | $\{(\infty,0),(\infty,0),(\infty,0),(\infty,0),(\infty,0),(10000,0)\}$ |
| $\tau_2$ | 500 | 200 | $\infty$ | 2 | $\{(\infty,0),(\infty,0),(\infty,0),(\infty,0),(\infty,0),(10000,0)\}$ |
| $\tau_3$ | 600 | 300 | $\infty$ | 3 | $\{(\infty,0),(\infty,0),(\infty,0),(\infty,0),(5000,0)\}$ |
| $\tau_4$ | 150 | 114 | 350 | 1 | $\dot{\Theta}^{+}_{\tau_1}$ |
| $\tau_5$ | 150 | 114 | 1000 | 2 | $\dot{\Theta}^{+}_{\tau_2}$ |
| $\tau_6$ | 150 | 114 | 3000 | 3 | $\dot{\Theta}^{+}_{\tau_3}$ |
| $\tau_7$ | 150 | 114 | 10000 | 4 | $\{(10000,0)\}$ |
| $\tau_8$ | 150 | 114 | 10000 | 5 | $\{(5000,0)\}$ |
| $\tau_9$ | 150 | 114 | 10000 | 6 | $\{(5000,0)\}$ |
| $\tau_{10}$ | 150 | 114 | 4000 | 7 | $\{(2000,0)\}$ |
| $\tau_{11}$ | 150 | 114 | 10000 | 8 | $\{(10000,0)\}$ |
| $\tau_{12}$ | 150 | 114 | 5000 | 9 | $\{(5000,0)\}$ |
| $\tau_{13}$ | 150 | 114 | 5000 | 10 | $\{(5000,0)\}$ |

**(b)** Properties

**Figure 2:** Distributed system example

| | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ | $\tau_5$ | $\tau_6$ | $\tau_7$ | $\tau_8$ | $\tau_9$ | $\tau_{10}$ | $\tau_{11}$ | $\tau_{12}$ | $\tau_{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $r^+(\tau)[\mu s]$ | 3000 | 7500 | 16000 | 300 | 1300 | 3150 | 4650 | 4800 | 5100 | 5700 | 6300 | 6450 | 6600 |
| $r^+_{les}(\tau)[\mu s]$ | 3000 | 7500 | 16000 | 300 | 700 | 1500 | 2100 | 2400 | 2850 | 3300 | 4050 | 4650 | 4650 |
| Red. [%] | 0 | 0 | 0 | 0 | 46,16 | 52,39 | 54,84 | 50 | 44,22 | 42,11 | 35,72 | 27,91 | 29,55 |
| R.n.P | 0 | 0 | 0 | 0 | 4 | 11 | 17 | 16 | 15 | 16 | 15 | 12 | 13 |

**Table 1:** Results of the example

All messages have a best-case execution time of 114 $\mu s$ which is the transmission of 1 byte payload and a worst-case execution time of 150 $\mu s$ which is 3 byte payload. CPU1 has three tasks which all transmit massages on the bus. Task $\tau_1$ transmits message $\tau_4$, $\tau_2$ transmits message $\tau_5$ and $\tau_3$ transmits message $\tau_6$. We need to consider the tasks on CPU1 in detail to compute the limitations. The tasks produce an initial burst on the CAN bus which leads to an enormous peak load on the bus. We use two limiting event streams to describe the competition based dependency in the system and to relax this peak load. With these dependencies we can bound the maximum load on the bus as CPU1 has not the capacity to produce all events at once.
We have analyzed the system with the limiting event streams and without the limiting event streams. In table 1 the improvements of the worst-case response times of the tasks can be seen. From message $\tau_4$ to message $\tau_{13}$ we have significant improvements of the response times. Especially, task $\tau_7$ has been improved by 54,84%. In the last line the reduced number of preemptions (R.n.P) is described. For example, task $\tau_{11}$ is preempted 15 times less if the limiting event streams are considered. The relaxation is caused by the two limiting event streams $\overline{\Theta}_1$ and $\overline{\Theta}_2$. In figure 3 the relaxation of the two outgoing event streams $\dot{\Theta}^+_{\tau_1}$ and $\dot{\Theta}^+_{\tau_2}$ is depicted. It can be observed that for the first 20 events an enormous relaxation can be achieved.
This leads to the conclusion that the consideration of this dependency can lead to an enormous reduction of the worst-case response times in the system. Certainly, there are also many cases in which the dependency does not lead to an improvement. These are the cases where the capacity of the resource is sufficient to produce the events. Therefore it is desirable to have a metric which can decide whether it is useful to consider the dependency or not.

## 7. Conclusion

In this paper we have introduced an new kind of dependency between tasks to improve the bounds of the response times in distributed real-time systems. The new dependency has been exemplarily

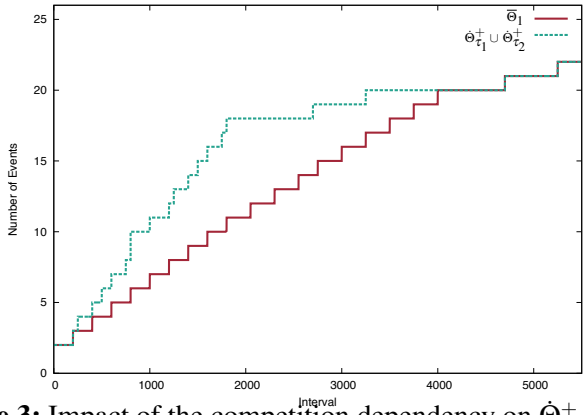



**Figure 3:** Impact of the competition dependency on $\dot{\Theta}^+_{\tau_1}$ and $\dot{\Theta}^+_{\tau_2}$

applied in an example to a fixed-priority system but is in general applicable to any scheduling policy. The idea is to determine if a resource has enough capacity to produce a specific number of events within a specific time interval. If the capacity is not available, the density of the events can be relaxed and therefore also the worst-case response times in the system. We have shown that the new competition based dependency can lead to significant improvements of the worst-case response times.

## References


[AS04] Albers, Karsten and Frank Slomka: *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*, pages 187–195. IEEE, July 2004, ISBN 0-7695-2176-2.

[GH98] Gutierrez, J. C. Palencia and Michael Gonzalez Harbour: *Schedulability analysis for tasks with static and dynamic offsets*. In *RTSS*, page 26 ff, 1998.

[GH99] Gutierrez, J. C. Palencia and Michael Gonzalez Harbour: *Exploiting precedence relations in the schedulability analysis of distributed real-time systems*. In *IEEE Real-Time Systems Symposium*, pages 328–339, 1999.

[Gre93] Gresser, Klaus: *An event model for deadline verification of hard real-time systems*. In *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, 1993.

[GRS96] George, L., N. Rivierre, and M. Spuri: *Preemptive and non-preemptive real-time uniprocessor scheduling*. Technical report, INRIA, 1996.

[HE05] Henia, Rafik and Rolf Ernst: *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*, pages 480–485, 2005.

[HE06] Henia, Rafik and Rolf Ernst: *Improved offset-analysis using multiple timing-references*. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 450–455, 2006.

[KAS08] Kollmann, Steffen, Karsten Albers, and Frank Slomka: *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.

[KPK+10] Kollmann, S., V. Pollex, K. Kempf, F. Slomka, M. Traub, T. Bone, and J. Becker: *Comparative application of real-time verification methods to an automotive architecture*. In *Proceedings of the 18th International Conference on Real-Time and Network Systems*, 2010.

[KPKS10] Kollmann, Steffen, Victor Pollex, Kilian Kempf, and Frank Slomka: *A scalable approach for the description of dependencies in hard real-time systems*. In *proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, 2010.

[KPS10] Kollmann, Steffen, Victor Pollex, and Frank Slomka: *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.

[Leh90] Lehoczky, John P: *Fixed priority scheduling of periodic task sets with arbitrary deadlines*. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, December 1990.

[PL05] Pellizzoni, Rodolfo and Giuseppe Lipari: *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*, pages 66–75, 2005.

[RE10] Rox, Jonas and Rolf Ernst: *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.

[Red04] Redell, Ola: *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)*, pages 239–248, 2004.

[Ric05] Richter, Kai: *Compositional Scheduling Analysis Using Standard Event Models - The SymTA/S Approach*. PhD thesis, University of Braunschweig, 2005.

[TC94] Tindell, Ken and John Clark: *Holistic schedulability analysis for distributed hard real-time systems*. Microprocessing and Microprogramming, 40:117–134, April 1994.

[Tin94] Tindell, Ken: *Adding time-offsets to schedulability analysis*. Technical report, University of York, Computer Science Dept, YCS-94-221, 1994.