Effects of Simultaneous Stimulation on the Event Stream Densities of Fixed-Priority Systems

Steffen Kollmann, Karsten Albers and Frank Slomka
Ulm University
{firstname}.{lastname}@uni-ulm.de

***Abstract*—In this paper we present an approach to calculate the maximum density of events in a distributed hard real-time system having tree-shaped dependencies. Thereby we will show that the density of events can be relaxed when taking scheduling dependencies into account. This relaxation has a direct impact on successive tasks and leads to tighter bounds for real-time analysis. We will also provide the real-time analysis for systems with static priorities.**

I. Introduction

Nowadays, embedded systems surround us in many day to day life situations. Steering a car, calling with a mobile phone or flying with a plane are only some examples where embedded systems help us. Thereby, the requirements for the systems increase from generation to generation.

Reliability, performance and costfactors are very important during the design process of such systems. The systems must also fulfill real-time constraints, since they are integrated in a technical context. Due to the increasing complexity of these systems makes it necessary to have appropriate methods for the real-time verification.

Many approaches have been developed to analyze hard real-time systems but most of them assume that the tasks in the system are independent. To include dependencies in the analysis leads us to a more accurate analysis of these systems. This has a direct effect on the performance of these systems, because more tasks can be scheduled on the same processor or a slower processor could be used. In this paper we introduce an approach to include dependencies of such tasks into the real-time analysis.

The rest of the paper is organized as follows. Chapter II gives an overview about related work in this domain. In chapter III we describe the problem we have explored. In chapter IV we will introduce our model based on the event stream model by K. Gresser [4]. Chapter V shows how a real-time analysis is possible with the model. On the basis of this model and the analysis we will show how to calculate the event streams in a heterogeneous distributed system. Subsequently, we give a case study showing the improvements of the approach. The conclusion follows at the end.

II. Related Work

For a real-time analysis it is necessary to choose an appropriate model. Most of the work in this area is based on the periodic task model with jitter as it is used by Tindell and Clark [11], for example. Many approaches have been developed based on this model but the simplicity of these models leads to a loss of accuracy. Another disadvantage is that no dependencies have been considered. Hence, the transaction model [10] has been developed where it is possible to group dependent tasks in transaction groups describing dependencies with fixed offsets. This approach has been integrated in the Holistic Scheduling Analysis. Redell generalized this approach in [9] and allows more sophisticated task chains. In his approach it is possible that a task can trigger more than one task what is in our opinion a more realistic case. The idea was used in [5] by Henia et al. This paper explores time-correlations between tasks via tree-shaped-dependencies. This allows to describe additionally relative dependencies to the last common predecessor of tasks. Henia et al. have improved this idea in [6].

Another approach to relax the density of events in a distributed system was explored in [7]. In this paper a correlation between the incoming jitter and the response time of a task was developed. This has led to a more relaxed output jitter which has a direct influence on the response time of successive tasks.

In this paper we use a correlation between tasks triggered by the same source to improve the density of events. For this purpose we use the event stream model [4] to describe the stimulation. The advantage of this model is that we are able to describe any kind of stimulation. Due to the simplicity we use the event streams, although the approach is extendible on more expressive hierarchical events streams [1].

III. Problem Formulation

In this paper we consider dependencies as shown in figure 1.

It shows three resources. The tasks τ_1, τ_2 and τ_3 are triggered by the same source which is described by an event stream Θ_A. Task τ_1 has got the highest priority, τ_2 the middle

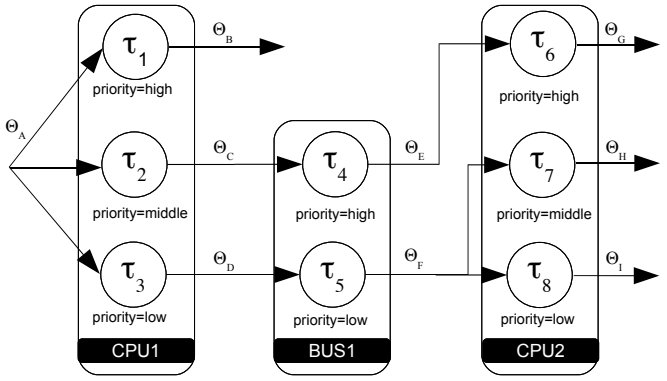

Fig. 1. Example of a distributed system where tasks have the same source.

and τ_3 the lowest. The tasks τ_2 and τ_3 send their information over a bus to a second processing element. We assume a priority driven bus. This can be modeled like a processing element where each channel over the bus can be considered as a task. The execution times of the tasks specify the time needed by the communication. The tasks τ_4 and τ_5, where τ_4 has got a higher priority than τ_5, trigger the tasks τ_6, τ_7 and τ_8 on $CPU2$. Task τ_6 has got the highest priority, τ_7 the middle and τ_8 the lowest. Note, τ_7 and τ_8 have got the same source.

In previous approaches it was necessary to consider the stimulation of task τ_3 independent of the stimulation of task τ_1 and τ_2 in order to calculate the outgoing event stream Θ_D of τ_3. The dependency that all three tasks are triggered by the same event stream and must occur therefore simultaneously, has not been considered in these analysis. Taking this dependency into account, may allow to reduce the resource $BUS1$ or $CPU2$ and still keeps all real-time constraints. Furthermore a response time reduction for successive tasks, τ_8 for example, is possible.

IV. Model

In this section we introduce our models. We differentiate between the task model and the model for the stimulation. The reason for this can be observed, for example, in the periodic task model with jitter [11] where we have got on the one hand the task model with the best case and worst case execution times, and on the other hand the model for the stimulation with jitter and period. But this is not enough to describe a major range of stimulations. Therefore we use the more generalized event stream model [4] to describe the stimulation.

A. Task Model

Γ is the set of tasks on one resource $\Gamma = \{\tau_1, ..., \tau_n\}$. A task is a 4-tuple with $\tau = (c, b, \rho, \Theta)$. c is the worst case execution time, b is the best case execution time, ρ is the priority for the scheduling (the lower the number the higher the priority) and Θ defines the stimulation of the task by an event stream. Let τ_{ij} be the j-th job/execution of task τ_i.

In our model we assume that a task can only generate an event at the end of its execution to notify other tasks. Furthermore we assume a fixed-priority scheduling.

B. Event Streams

Event streams have been first defined in [4]. The purpose was to give a generalized description for every kind of stimuli. The basic idea is to define an event function $E(I, \Theta)$ which can calculate for every interval I the maximum amount of events occurring within I. In the following, when speaking of intervals 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 is to notate for each number of events the minimum interval which can include this number of events. Therefore we get an interval for one event (which is infinitely small and therefore considered to be zero), two events and so on. The result is a sequence of intervals showing a non-decreasing behaviour. The reason for this behaviour is, that 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. This sequence of intervals shows a periodic behaviour and is called event stream. Each of the single intervals is called event stream element.

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*.*

Each event stream element describes a set of intervals of the sequence. For the event stream element θ the interval $a + k \cdot p$ is part of the sequence and all the intervals with $k \in N$. An event stream models a given sequence if all the elements and only the elements of the sequence can be generated using the event stream elements. Therefore it is possible to calculate for each interval the maximum amount of events that can occur within this interval:

Event Stream Function:

$$E(I,\Theta){=}\sum_{\theta\in\Theta} E(I,\theta); E(I,\theta){=}\begin{cases} 0 & I{<}a_\theta \\ \left\lfloor \frac{I-a_\theta}{p_\theta}{+}1 \right\rfloor & I{\geq}a_\theta \wedge p_\theta{<}\infty \\ 1 & I{\geq}a_\theta \wedge p_\theta{=}\infty \end{cases} \quad (1)$$

As inverse function we define the following function which gives to a number of events the minimum interval in which these events can occur:

Request Time Function:

$$RT(n,\Theta){=}min\{I|E(I,\Theta){=}n\} \quad (2)$$

Furthermore the event stream model complies the characteristic $E(I_1+I_2,\Theta) \leq E(I_1,\Theta)+E(I_2,\Theta)$. This characteristic is called sub-additivity. This means that the maximum number of events of an interval cannot exceed the cumulated maximum number of events of its subintervals.

Events can occur in a greater distance than it is described in the event stream which describes only the minimum distance between a number of events.

An event stream in which all elements have either the same or an infinite period is called homogeneous and every event stream can be made homogeneous using the least common multiplier of its periods as new period of all elements to complete its set of event stream elements. With a infinite (∞) period it is possible to model irregular behaviour.

Note that the order of the elements is of no concern for the evaluation. For the purpose of evaluation it is not necessary to find the exact minimum intervals. It is sufficient to find a lower bound for all intervals. This can allow to simplify the event stream (also might mean to accept an overly pessimistic description). A detailed definition of the concept and the mathematical foundation can be found in [2].

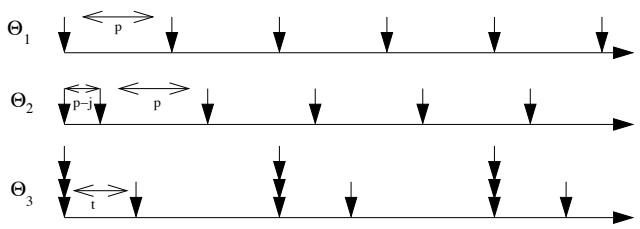

Fig. 2. This figure shows three different event sequences. p describes a period, t and j an offset

In figure 2 some examples for event streams can be found. The first one Θ_1 = (p,0) has a strictly periodic stimulus with a period p. The second example Θ_2 = (∞,0), (p,p-j) shows a periodic stimulus in which the single events can jitter within a jitter interval of size j. In the third example Θ_3 = (p,0), (p,0) , (p,0), (p,t) three events occur at the same time and the fourth occurs after a time t. This pattern is repeated with a period of p. Event streams can describe all these examples in an easy and intuitive way.

C. Normalized Event Streams

Some event streams can have several different descriptions in the event stream model. For an efficient implementation of the approach in chapter VI we define a normalized event stream. We introduce the identifier θ_i denoting the i-th element in the event stream.

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

Here normalization means that all aperiodic events are described first in the stream. All other events have the same period and follow directly after the aperiodic events. This is no restriction, because every event stream can be transformed into this pattern. To achieve this we calculate the least common multiplier for all periods and use it as new period. We shift the offset of all periodic events until the equation is fulfilled. The shift operation is performed as follows: $(p,a) = (\infty,a),(p,(a+p))$. Furthermore it is necessary that the aperiodic part is sorted by its offset as well as the periodic part.

It is useful to define names for certain properties of the event stream. N is the number of tuple in the event stream. N_∞ is the number of tuples with a period $p = \infty$ and N_p is the number of tuples with a period $p \neq \infty$.

V. Real-Time Analysis

The most usual way to do a real-time analysis is to perform a response time analysis as introduced by Lehoczky et. al. [8]. The condition $\forall\tau \in \Gamma : WCRT_k(\tau) \leq d_\tau$ holds when the real-time analysis is successful. In order to calculate the worst case response time we have adapted the approach from [8].

$$WCRT_k(\tau)=min\{I|I=k\cdot c_\tau+\sum_{\tau'\in HP} E(I,\Theta_{\tau'})\cdot c_{\tau'}\} \tag{4}$$

The equation is similar to the common definition of the worst case response time. Only the calculation of the influence of higher priority events has been changed. The amount of execution produced by higher priority tasks can be calculated by the event function multiplied by the worst case execution time. By means of a fixed point iteration the worst case response time can be calculated for every k.

Our approach is also applicable to other analysis, for example the analysis described by S. Baruah [3].

VI. Worst Case Density

In this chapter we present a technique to calculate the event streams in a distributed system by considering dependencies caused by same sources. Thereby the assumptions made in chapter IV are used.

A. Improvement by Simultaneous Events

As mentioned in chapter II and III it is important to consider dependencies during an analysis, in order to relax worst case scenarios. One of such dependencies is depicted in figure 3. Note, that $c_{i,j}$ is the worst case execution time of the j-th job

of task τ_i and $b_{i,j}$ is the best case execution time of the j-th job of task τ_i.

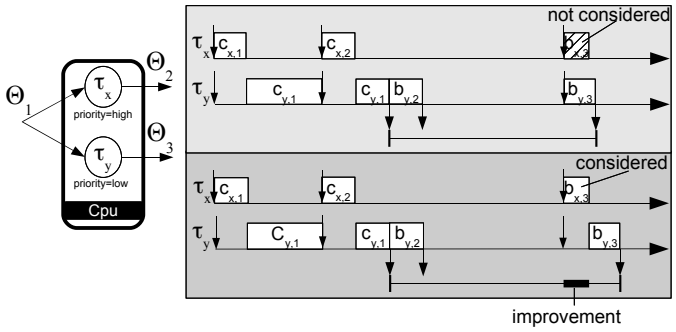

Fig. 3. Improvement of the event density by considering simultaneous stimulation

In the left part of the figure we see two tasks scheduled on the same cpu. Both are triggered by the incoming event stream called Θ_1. For successive tasks which could be triggered by τ_x and τ_y it is important to calculate the outgoing event streams Θ_2 and Θ_3 as accurate as possible. We assume, as it is common in real-time analysis, that a task generates an event at the end of its execution.

In the right part of the picture the scenario is depicted if no dependencies and if dependencies are considered. The above gantt-diagramm shows how the outgoing event stream of τ_y can be determined if no dependencies are considered. The first event occurs as late as possible and all further events as early as possible. This leads to the maximum density of events. (The proof for this is given in the next subsection). Note, that only executions of the higher priority task are considered when the first event is delayed. This means, that no further executions of τ_x are considered during other executions of τ_y. Such kind of calculation is the common proceeding in real-time analysis. But this is only an estimation as it can be seen in the lower gantt-chart. Since the task are triggered by the same event stream the execution of the higher priority task has to be considered during every execution of the lower priority task. This leads to more relaxed outgoing event streams. In the lower gantt-chart of the figure it can be seen that the occurrence of the third outgoing event can be relaxed by considering the best case execution of the higher priority task. Consequently, we have to improve the existing analysis technique in a way that such scenarios can be considered.

B. Calculation of the Density of Events

In order to calculate the outgoing event stream of a task we have to determine the maximum density of events that a task can produce. The next lemma gives the worst case. We assume that the task is triggered by the maximum density of events which is described by the event stream. The event stream stimulating a task is called *incoming event stream* and the event stream produced by a task is called *outgoing event stream*.

Lemma 1: *A number of outgoing events occur in the maximum density when the first event is delayed as much as possible and all further events occur as early as possible.*

Proof: We assume that two outgoing events e_1 and e_2 exist having a higher density than the events fulfilling the assumption. If e_1 and e_2 are closer together than in the assumption, this would mean either e_1 arrives later than allowed by the assumption or e_2 arrives earlier than allowed by the assumption. This is a contradiction, because we assume already the maximum or minimum values for both arrival times. So there must be two other events later in the outgoing event stream having a shorter distance to each other. Assume that two events are occurring closer than in the assumption and the first event is delayed as much as possible and the second arrives as early as possible, this would mean that the corresponding incoming events also have a shorter distance to each other than the first two incoming events. But this is in contradiction to the event stream definition. The proof for another number of events is analog. □

This lemma is similar to other techniques. Tindell, for example, uses this in [11] to determine the worst case response time of tasks in a distributed system. There the maximum jitter for the first event is assumed and all further events occur within the minimal period. This leads finally to the maximum density of events.

We are able to determine the outgoing event stream of a task via lemma 1. It is shown in figure 4 that three tasks τ_1, τ_2 and τ_3 are scheduled on the same resource. Thereby, τ_1 has a higher priority than τ_2 and τ_2 has got a higher priority than τ_3. τ_2 and τ_3 are triggered by the same source whereas τ_1 is triggered by a different one. Arrows over the timeline show requests (**R**equest **T**ime; RT) for a task and arrows under the timeline show when a task produces an outgoing event. The points in time l_1 up to l_6 are called Request End Time (**R**equest **E**nd **T**ime; RET). This is the time interval between when a task produces an event and t_0.

For the calculation of the density of the outgoing events we define an extended interval function. This function gives for an amount of events the minimum interval in which they can occur. We call it interval dependency function and define it as follows:

$$I_D(n,\tau)=\begin{cases}0 & n=1\\ RET(n,\tau)-RET(1,\tau) & n>1\end{cases} \quad (5)$$

According to the event stream definition one event occurs

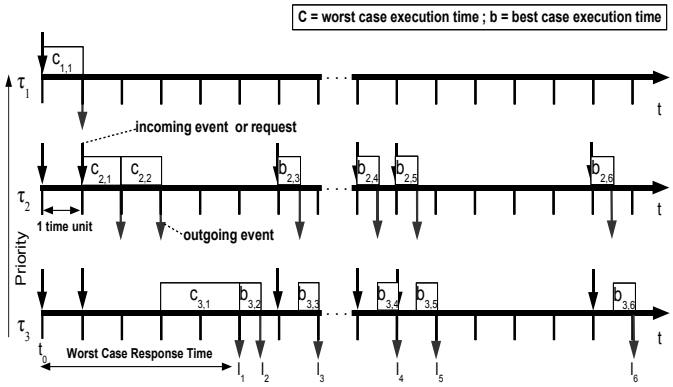

Fig. 4. Scheduling example of the tasks τ_1, τ_2 and τ_3

always in the interval zero. Hence, we distinguish in the equation 5 between two cases. The first case describes the interval for one event which is always zero according to the event stream definition. All other events are covered by the second case via the Request End Times. Since there are a lot of cases to distinguish we define the following request end time function:

$$RET(n,\tau)=\begin{cases} WCRT_1(\tau) & n=1 \\ RT(n,\Theta_\tau)+b_\tau+HP(\tau) & n>1 \ \wedge \ RET(n-1,\tau)\leq RT(n,\Theta_\tau) \\ RET(n-1,\tau)+b_\tau & n>1 \ \wedge \ RT(n,\Theta_\tau) \ < \ RET(1,\tau) \\ RET(n-1,\tau)+b_\tau+HP(\tau) & n>1 \ \wedge \ RET(1,\tau)\leq RT(n,\Theta_\tau) \ \wedge \\ & RT(n,\Theta_\tau)<RET(n-1,\tau) \end{cases} \tag{6}$$

The equation 7 describes the maximum amount of executions which higher priority tasks scheduled by the same source can contribute to the calculation.

$$HP(\tau)= \sum_{\substack{\tau' \in \Gamma \\ \Theta'_\tau=\Theta_\tau \\ \rho'_\tau<\rho_\tau}} b_{\tau'} \tag{7}$$

The different cases of this equation are visualised in figure 4. According to lemma 1 the first event is delayed by its worst case response time of its first instance. In figure 4 $\tau_{3,1}$ is delayed up to l_1.

The second case covers the arrival of the next job after the request end time of the previous job. In this case, visualised by job $\tau_{3,3}$, the request is only delayed by its execution time (best case) and the execution time of the higher priority jobs triggered by the same event and therefore arriving simultaneously with the considered job.

In the third case, applying for job $\tau_{3,2}$, the new job arrives before the first job is finished. In this case the RET of the previous job includes already the execution time of all higher priority jobs, so only the execution of the job itself has to be added.

In the last case the job arrives somewhere between the RET of the first job and the RET of its previous job. This case is visualised in figure 5. The $RET(2,\tau_2)$ of the second job consists of the worst case response time of the first job and $b_{2,2}$. Note that the execution time of the second job of the first task is already included in the worst-case response time of the first job. The third event and therefore $b_{2,3}$ is not considered at this stage. Therefore we have to add again the execution time of the job and all concurrently arriving higher priority jobs, to achieve the request end time of job $\tau_{2,3}$.

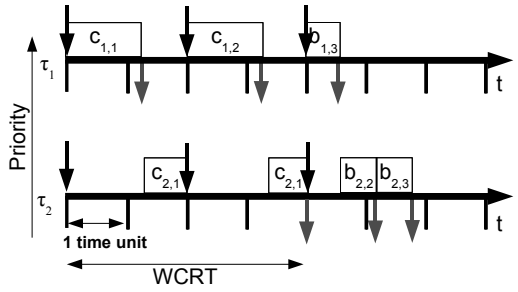

Fig. 5. Scheduling Example of the tasks τ_1 and τ_2

By means of the equation 5 and 6 it is possible to determine for every count of events the minimum interval in which they occur. This means the maximum density of the outgoing event stream can be determined.

We will give two scenarios to show the calculation of the intervals. In both scenarios we assume the architecture from figure 1. We calculate the outgoing event stream Θ_I, whereas the event streams Θ_E and Θ_F and the parameters of τ_6, τ_7 and τ_8 are given.

Scenario 1: Assume that Θ_E $=\{(12,0)\}$, Θ_F $=\{(20,0),$ $(20,2),(20,12)\}$, τ_6 $=(2,2,1,\Theta_E)$, τ_7 $=$ $(2,1,2,\Theta_F)$ and τ_8 $=(4,1,3,\Theta_F)$. With these parameter we get a schedule like in figure 4. First we determine the interval for one event which is $I_D(1,\tau_8) = 0$ according to case one equation 5. For two events we have to consider the second case from equation 5. There we must determine the request end time for the first event and for the second event. According to equation 6 $RET(1,\tau_8) = WCRT_1(\tau_8) = 10$. The request end time of the second event can be determined by the third case of equation 6. Since the count of events is greater than one and $RET(1,\tau_8) = 10 > 2 = RT(2,\Theta_F)$ and the second request occurs within the worst case response time $RET(1,\tau_8) =$ $10 > 2 = RT(2,\Theta_F)$, we can calculate the request end time by the request end time of $RET(1,\tau_8) = 10$ and the best case execution of τ_8 $(b_{\tau 8} = 1)$. The result for the second interval

is $I_D(2,\tau_8) = 10 - 11 = 1$. The interval for three events can be calculated by the second case of equation 6. Since the $RET(2,\tau_8) = 11 \leq 12 = RT(3,\Theta_F)$, we can determine the request end time of the third request by the sum of the request time of the third request $RT(3,\Theta_F) = 12$ plus the best case execution of τ_8 ($b_{\tau 8} = 1$) plus the best executions of all higher priority tasks triggered by the same source $HP(\tau_8) = 1$. The result is $12 + 1 + 1 = 14$. The result of $I_D(3,\tau_8)$=4. The calculation of the remaining intervals is similar to the last.

Scenario 2: Assume that Θ_E $=\{(\infty,0),(250,210)\}$, $\Theta_F = \{(\infty,0),(250,140)\}$, $\tau_6 = (50,40,1,\Theta_E)$, $\tau_7 = (40,30,2,\Theta_F)$ and $\tau_8 = (50,50,3,\Theta_F)$. With these parameter we get a schedule like in the case study chapter VII. First we determine the interval for one event which is $I_D(1,\tau_8) = 0$ according to case one of equation 5. For two events we have to consider the second case from equation 5. There we must determine the request end time for the first event and for the second event. According to equation 6 is $RET(1,\tau_8) = WCRT_1(\tau_8) = 140$. The request end time of the second event can be determined by the second case of equation 6. Since the $RET(1,\tau_8) = 140 \leq 140 = RT(2,\Theta_F)$, we can determine the request end time of the second request by the request time of the second request $RT(2,\Theta_F) = 140$ plus the best case execution of τ_8 ($b_{\tau 8} = 50$) plus the best executions of all higher priority tasks triggered by the same source $HP(\tau_8) = 30$. The result is $140 + 50 + 30 = 220$. The result of $I_D(2,\tau_8)$=220-140=80. The calculation of the remaining intervals is similar to the last case.

C. Resulting Event Streams

After we have shown how to calculate the request end times, we will show now how to achieve the outgoing event stream. The main disadvantage of the previous described functions is that the interval function is very difficult to calculate for arbitrary event streams. For this reason we have defined the normalized event streams in section IV. By means of this normalization we can very easily run through the event stream and calculate the RET. As mentioned in chapter IV the normalization is without lost of generality.

Equation 8 defines how the incoming event stream is processed in order to calculate the outgoing event stream.

$$\Theta_{out} = \left\{ \bigcup_{i=1}^{j} \begin{pmatrix} \infty \\ RET(i,\tau) - WCRT_1(\tau) \end{pmatrix}^T, \bigcup_{i=j+1}^{N_P+j} \begin{pmatrix} p_i \\ RET(i,\tau) - WCRT_1(\tau) \end{pmatrix}^T \right\} \tag{8}$$

The equation consists of two parts. The first part gives all event stream elements having an aperiodic behaviour and the second part all event stream elements having a periodic behaviour. The offset for the i-th element of both parts is the distance between that point in time when the i-th event of the incoming event stream has been processed and that point in time when the first event of the incoming event stream has been processed. The following incoming events have no influence on the offset of the i-th event, because these events can occur time-shifted according to the event stream definition.

The division of periodic and aperiodic outgoing events is determined by a gap of time. A gap occurs when the processing of the i-th event finishes before the arrival of the (i+1)-th event. We need the time of the first gap after the finishing of all aperiodic incoming events. At this point it can be assumed that the aperiodic behaviour is finished, because the remaining events have a constant execution time and occur periodically. Equation 9 shows how to calculate this point in time.

$$j = min\{\forall i : RET(i,\tau) \leq RT(i+1,\Theta_\tau) \wedge RT(i+1,\Theta_\tau) \geq min\{a_l | p_l \neq \infty\}\} \tag{9}$$

As mentioned, the main problem is to determine the request time efficiently. By means of the normalized event streams we are able to calculate these points in time. Equation 10 delivers for the i-th event of a normalized event stream the time when it is requested. By means of the modulo calculation in the second case it is possible to ensure that aperiodic events are to be considered only once. Note, we use the definition from chapter IV.

$$RT(n,\Theta) = \begin{cases} a_i & i \leq N \\ \left\lfloor \frac{i-N_\infty}{N_P} \right\rfloor \cdot p_{((i-N) mod\ N_P+N_\infty)} + a_{((i-N) mod\ N_P+N_\infty)} & i > N \end{cases} \tag{10}$$

A short example will make the calculation clear. Assume a normalized event stream $\Theta = \{(\infty,0),(\infty,10),(\infty,20),(150,50),(150,70),(150,90)\}$. For the first N events the calculation is clear, since the elements are sorted by their offsets. So the first event occurs at 0, the second at 10 and so on. Assume that we would like to determine the request of the 14th event. This can be calculated as follows ($N = 6, N_\infty = 3, N_P = 3$):

$$\left\lfloor \frac{14-3}{3} \right\rfloor \cdot p_{((8) mod\ 3+3)} + a_{((8) mod\ 3+3)} = 520$$

Here we give two further scenario in order to show how the outgoing event stream can be calculated. We use the parameters from scenario 1 and 2.

Scenario 3: Assume that $\Theta_E = \{(12,0)\}$, $\Theta_F = \{(20,0),(20,2),(20,12)\}$, $\tau_6 = (2,2,1,\Theta_E)$, $\tau_7 = (2,1,2,\Theta_F)$ and $\tau_8 = (4,1,3,\Theta_F)$. In order to determine the outgoing event

stream we use equation 8. The gap we have to find by equation 9 can be determined during the calculation. We calculate the first tuple of the event stream by the substraction of the worst case response time $WCRT_1(\tau_8) = 10$ and the request end time of the first request. According to 6 is this the worst case response time. Subsequently, the first tuple is $(\infty, 0)$. Before we calculate the second tuple we have to verify whether a periodic behaviour starts or not. According to equation 9 this is not the case, because the $RET(1, \tau_8) = 10 > 1 = RT(2, \Theta_F)$. So we determine the next tuple by calculating $RET(2, \tau_8)$. This is the $RET(1, \tau_8) = 10$ plus the best case execution of τ_8 ($b_{\tau 8} = 1$). So the next tuple is $(\infty, 1)$. Now we verify again whether a gap exist. This time there exist a gap and the next request occur periodically. So we can assume that all further events occur with a period. The $RET(3, \tau_8) = 12 + 1 + 1 = 14$. The resultant tuple is (20,4). Only two more tuple have to be considered to fulfill equation 8. The remaining RET can be calculated by the second case of equation 6. So $RET(4, \tau_8) = 12$ and $RET(4, \tau_8) = 14$. The result of the outgoing event stream is $\Theta_I = \{(\infty, 0)(\infty, 1)(20, 4)(20, 12)(20, 14)\}$

Scenario 4: Assume that $\Theta_E = \{(\infty, 0), (250, 210)\}$, $\Theta_F = \{(\infty, 0), (250, 140)\}$, $\tau_6 = (50, 40, 1, \Theta_E)$, $\tau_7 = (40, 30, 2, \Theta_F)$ and $\tau_8 = (50, 50, 3, \Theta_F)$. In order to determine Θ_I we use equation 8. The gap we have to find by equation 9 can be determined during the calculation. We calculate the first tuple of the event stream by the substraction of the worst case response time $WCRT_1(\tau_8) = 140$ and the request end time of the first request. According to 6 is this the worst case response time. Subsequently, the first tuple is $(\infty, 0)$. Before we calculate the second tuple we have to verify whether a periodic behaviour starts or not. According to equation 9 this is the case. The $RET(1, \tau, \Theta_F) = 140 \leq 140 = RT(2, \Theta_F)$ and the next request occurs periodically. We found our gap and can use the second part of equation 8. Since $RET(1, \tau, \Theta_F) = 140 \leq 140 = RT(2, \Theta_F)$ we use the second case of equation 6, the request end time of $RET(2, \tau_8, \Theta_F) = RT(2, \Theta_F) + b_{\tau 8} + HP(\tau_8) = 220$. Subsequently we get as new tuple $(250, 80)$. The resulting event stream $\Theta_I = \{(\infty, 0)(250, 80)\}$.

VII. Case Study

In order to show the improvements of our approach we have carried out a case study. In this we explore a hypothetical distributed system and compare the density of events with and without the introduced method.

We consider again the example depicted in figure 1 whose behaviour was described in chapter III. In order to perform the analysis we have to define the parameters of the system which are described in the next tables.

CPU1	τ_1	τ_2	τ_3
c	50	60	80
b	40	50	50
ρ	1	2	3
Θ	Θ_A	Θ_A	Θ_A

Bus1	τ_4	τ_5
c	40	40
b	20	20
ρ	1	2
Θ	Θ_C	Θ_D

CPU2	τ_6	τ_7	τ_8
c	50	40	50
b	40	30	50
ρ	1	2	3
Θ	Θ_E	Θ_F	Θ_F

TABLE I
PARAMETERS OF THE DISTRIBUTED SYSTEM WHICH IS DEPICTED IN FIGURE 1

With the given parameters we have calculated the event streams in the system. We use only normalized event streams for our calculation. The results for the event streams are shown in table III. The event streams have been calculated with and as well as without the approach.

Θ	with Approach	without Approach
Θ_A	{(250,0)}	{(250,0)}
Θ_B	{(∞,0),(250,240)}	{(∞,0),(250,240)}
Θ_C	{(∞,0),(250,230)}	{(∞,0),(250,190)}
Θ_D	{(∞,0),(250,200)}	{(∞,0),(250,110)}
Θ_E	{(∞,0),(250,210)}	{(∞,0),(250,170)}
Θ_F	{(∞,0),(250,140)}	{(∞,0),(250,50)}
Θ_G	{(∞,0),(250,200)}	{(∞,0),(250,160)}
Θ_H	{(∞,0),(250,80)}	{(∞,0),(∞,30),(250,240)}
Θ_I	{(∞,0),(250,80)}	{(∞,0),(∞,50),(250,120)}

TABLE II
RESULTS OF ALL EVENT STREAMS IN THE DISTRIBUTED SYSTEM. THE RESULTS ARE COMPUTED WITH AS WELL AS WITHOUT THE APPROACH.

The result shows that we have an improvement of the event density. Or in other words, the events occur not so dense in the system when the dependencies are considered. Let us consider, for example, the event stream Θ_E. Without the approach the density of two events is 170. With the approach on the other hand we have a density of 210. This is an improvement of 19%.

This relaxation has a direct influence on the real-time analysis. Since the events occur not so dense, the response time of certain tasks can be reduced.

In order to highlight the results a little bit more and to represent the influence on the real-time analysis, we take a closer look on the task τ_8 and the corresponding incoming and outgoing events.

In table III the reduction of the density of events caused

by the approach is shown. We consider here the event streams Θ_F, Θ_H and Θ_I.

n	Θ_F	Θ_F'	$Imp.$	Θ_H	Θ_H'	$Imp.$	Θ_I	Θ_I'	$Imp.$
1	0	0	0%	0	0	0%	0	0	0%
2	140	50	64,28%	80	30	62,5%	80	50	37,5%
3	390	300	23,07%	330	240	27,27%	330	120	63,63%
4	640	550	14,06%	580	490	15,51%	580	370	36,2%
5	890	800	10,11%	830	740	10,84%	830	620	25,3 %
6	1140	1050	7,89%	1080	990	8,33%	1080	870	19,44%
7	1390	1300	6,47%	1330	1240	6,76%	1330	1120	15,78%
8	1640	1550	5,48%	1580	1490	5,69%	1580	1370	13,29%
9	1890	1800	4,76%	1830	1740	4,91%	1830	1620	11,47%
10	2140	2050	4,20%	2080	1990	4,32%	2080	1870	10,09%

TABLE III
THIS SHOWS THE IMPROVEMENT OF THE APPROACH ON THE EVENT STREAMS Θ_F, Θ_H AND Θ_I. Θ SHOWS THE INTERVALS WITH THE DEPENDENCY, Θ' SHOWS THE INTERVALS WITHOUT THE DEPENDENCY, IMPROVEMENT IS GIVEN IN %

The table shows that the method has a very great effect on the first events. This is caused by the fact, that the periodic part of the events have no great influence on the intervals of the first events. So when the number of events rise and the periodic part of the events rise, the relaxation of the events goes down. But the absolute relaxation is constant. This behaviour is no disadvantage, since the greatest density of n-events is always at the beginning of the event stream. So the absolute relaxation of the first events is important, because there is a great effect on the real-time analysis. The influence of the approach on the event streams $\Theta_F, \Theta_H, \Theta_I$ are depicted in figure 6.

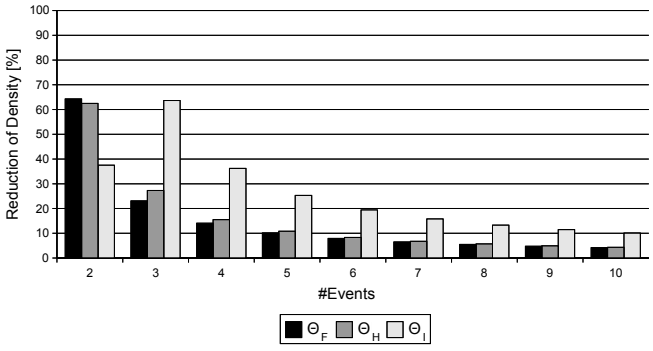

Fig. 6. This figure shows the improvements of the intervals in percent of the event stream Θ_F, Θ_H and Θ_I.

In this figure it can be seen that we have improvements up to 64% at the first two events.

This relaxation of the events has got a direct influence of the response of task τ_8. Without dependencies the response time of τ_8 is 230 and with dependencies is the response time 140. This leads to a reduction of 39.13% of the response time. At this point the effect of the approach can be seen. This study shows, that the consideration of such dependencies leads to tighter response times in real-time analysis.

VIII. CONCLUSION

The purpose of this paper was to calculate more precise event streams for the real-time analysis of fixed priority systems. This was possible by using the more general event stream model. We have given the calculation of outgoing event streams from tasks which are scheduled by fixed priorities. Thereby we have considered the dependencies caused by the scheduling on a processor. Especially the new consideration of the dependency that tasks can be triggered by the same source leads to a further relaxation.

We have also shown how a real-time analysis can be achieved with the result. This enables us to do a real time analysis for appropriate heterogeneous distributed systems. A further step is to use this approach in connection with the more expressive hierarchical event stream model [1].

REFERENCES

[1] Karsten Albers, Frank Bodmann, and Frank Slomka. Hierarchical event streams and event dependency graphs: A new computational model for embedded real-time systems. In *ECRTS '06: Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pages 97–106, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Karsten Albers 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.

[3] Sanjoy K Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24(1):93–128, 2003.

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

[5] Rafik Henia 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, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Rafik Henia 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, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[7] Rafik Henia, Razvan Racu, and Rolf Ernst. Improved output jitter calculation for compositional performance analysis of distributed systems. In *Proceedings Workshop on Parallel and Distributed Real-Time Systems*, March 2007.

[8] John P Lehoczky. 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.

[9] Ola Redell. 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, Washington, DC, USA, 2004. IEEE Computer Society.

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

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