Response Time Analysis of EDF Distributed Real-Time Systems

J.C. Palencia and M. González Harbour

Abstract[1]**—Offset-based response time analysis of tasks scheduled with fixed priorities has demonstrated to be a powerful tool to analyze many task models with different kinds of timing constraints, like regular periodic tasks, suspending tasks, distributed systems, tasks with varying priorities, multiframe models, etc. Offset-based analysis techniques are capable of performing a global schedulability analysis in distributed systems, as opposed to the less efficient techniques that consider each processing or communication resource as independent. In this paper we extend the offset-based schedulability analysis techniques to systems with EDF scheduling, using analytical techniques that are similar to those developed for fixed priority scheduling. With this new analysis, we now have a complete set of techniques to perform the analysis of different task models in distributed heterogeneous systems, i.e., processors and communication networks having either fixed priority or EDF schedulers.**

Index Terms—**Distributed Systems, EDF, Real-time Systems, Schedulability analysis.**

I. Introduction

RATE Monotonic Analysis (RMA) [1][3] allows an exact calculation of the worst-case response time of tasks in single-processor real-time systems, including the effects of task synchronization [10], the presence of aperiodic tasks, the effects of deadlines before, at or after the periods of the tasks [2] and tasks with precedence constraints in single processor systems [4]. For multiprocessor and distributed hard real-time systems there are fixed-priority schedulability analysis techniques, such as the Holistic analysis [12], which make an independent analysis on each processing or communication resource, and then iterate over this analysis. Because of the independent resource assumption, these techniques produce pessimistic results. Offset-based analysis techniques for fixed-priorities [13][8][9] make a global analysis of the distributed system and are able to eliminate much of the pessimism, obtaining response times that are significantly lower, and increasing the maximum schedulable utilization up to an additional 25%. Offset-based analysis techniques are also capable of correctly modeling and analyzing the effects of suspension in hard real-time tasks.

Spuri [6][7] adapted the Holistic Analysis technique to systems based on EDF schedulers. In this paper we extend Spuri's techniques and the offset-based analysis techniques developed for fixed-priority systems to analyze systems based on dynamic priorities, under EDF. This extension allows us to take advantage of the benefits that a global system-wide analysis has on improving the estimations of worst-case response times over the values obtained through the holistic analysis. Because offset-based analysis techniques for EDF are very similar to those for fixed priorities, by combining the two it is possible to analyze heterogeneous systems in which some nodes are scheduled under EDF while others are scheduled under fixed priorities.

The paper is organized as follows. In Section 2 we review the analysis technique derived by Spuri for periodic tasks under EDF, as the basis for our technique. Section 3 contains the analysis for tasks that share resources in mutual exclusion. Then, in Section 4 we obtain the analysis technique for EDF tasks with static offsets. In Section 5 we extend the analysis to handle tasks with dynamic offsets, and we show how to support different timing constraints under the task model with dynamic offsets. Section 6 shows the simulation results obtained with the new technique, comparing them with those of current techniques, based on independent tasks. Finally, in Section 7 we give our conclusions.

II. Analysis for Periodic Tasks

In this section we will consider a task model with periodic tasks scheduled under the EDF algorithm. In this model, the system is composed of a set of n periodic tasks executing in a single processor. Each task τ_i is activated periodically with a period of T_i, and has a worst-case execution time of C_i. Each task activation or instance is called a *job*, and must execute before a deadline d_i, relative to its activation time. Each task job can have a release jitter bounded by a maximum value J_i, so the release of the task may be delayed a maximum of J_i from its activation time. Both J_i and d_i may be greater than the task's period, in such a way that several activations of a task can be pending for execution at a given instant. The task set is scheduled under a preemptive EDF scheduler, that is, if some tasks

1. This work has been funded by the *Comisión Interministerial de Ciencia y Tecnología* of the Spanish Government under grant TIC 2002-04123-C03 and by the *Commission of the European Communities* under contract IST-2001-34140 (FIRST project)

The authors are with "Departamento de Electrónica y Computadores, Universidad de Cantabria, 39005-Santander, SPAIN". E-mail addresses: {palencij,mgh}@unican.es

are ready to execute, the scheduler will run the task with the earliest deadline, relative to the current time. For each task τ_i we define its worst-case response time as the maximum of the response times of all the jobs, each of which is defined as the difference between the job's completion time and its activation time. The worst-case response time will be called R_i.

In [6] Spuri developed a method to calculate the worst-case response times of tasks under this model. We will present here that analysis technique with some changes in notation that will facilitate the introduction of offsets in Section 4.

The response time analysis is based on the creation of the longest busy period. A busy period is defined for EDF scheduling as an interval of time during which the CPU is busy processing pending execution of any task. In fixed priority scheduling, the worst-case response time of a task τ_a is found after a critical instant, when the activation of τ_a coincides with the activation of all tasks with higher priority after having experienced the maximum jitter. In that situation, the critical instant coincides with the start of a busy period. In EDF scheduling that property is not true, but the busy period concept is still useful. The following theorem helps us to find the critical instant for a task:

Theorem 1. The worst-case response time of a task τ_a is found in a busy period in which all other tasks are released simultaneously at the beginning of the busy period, after having experienced their maximum jitter (i.e., each task τ_i with an activation J_i time units before the start of the busy period).

Proof: Let t_0 be the instant at which a task τ_i is activated the first time in the busy period, and let D be the deadline of an instance of the analyzed task, τ_a, relative to the beginning of the busy period. Suppose that t_0 does not coincide with the beginning of the busy period: in this circumstance, if we move the activation pattern of τ_i to occur earlier, down to the point when the first activation coincides with the beginning of the busy period, it is possible that new activations occur in the busy period, making it longer. The deadlines of each activation of τ_i will be earlier, so an activation with a deadline after instant D may have been moved to have a deadline before D, thus increasing the response time of task τ_a. On the other hand, if the first activation had experienced its maximum jitter but continues to be released at the start of the busy period, the following activations will occur the earliest possible and with a deadline that is earlier, relative to the beginning of the busy period. Therefore, increasing the jitter of the first activation can only increase the response time of task τ_a, and the theorem follows.❒

Note that, contrary to the other tasks, releasing the analyzed task at the start of the busy period may not lead to its worst-case response time. If we move the activation pattern of τ_a to occur earlier, we are causing deadline D to be earlier too, and this could imply that some deadlines of other tasks that previously occurred before D could now occur after D, and thus make the response time of task τ_a become smaller. So, the critical instant for a task is found in a busy period that is started by the simultaneous activation of all tasks except perhaps the one under analysis.

In order to calculate the worst-case response time of task τ_a, we will now calculate, under the conditions of theorem 1, the worst-case contribution of a task τ_i to a busy period of length t when the deadline of τ_a occurs at instant D. We will name this contribution $W_i(t,D)$. Figure 1 shows a scenario for calculating this contribution.

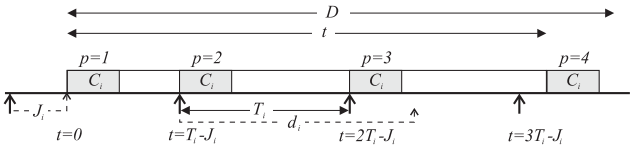

Fig. 1. Scenario for calculating the worst-case contribution

When we calculate the worst-case contribution of a task τ_i to a busy period we must consider the activations that occur in the interval $[0,t)$ but we must only consider the activations with deadline before or at D. Each of the activations will be identified with a sequence number p, starting at p=1. In Figure 1, activation p=4 occurs before t, but its deadline is after D, so under the EDF rules it must not be considered for the worst-case contribution.

To calculate the number of activations of τ_i in the busy period we can see that the identifier of the last activation in that busy period, p_t, is the only value of p that simultaneously satisfies:

$$(p-1)T_i - J_i < t \tag{1}$$

and:

$$pT_i - J_i \geq t \tag{2}$$

from which we get:

$$p < \frac{t+J_i}{T_i} + 1 \qquad \text{and} \qquad p \geq \frac{t+J_i}{T_i} \tag{3}$$

Given that p_t is an integer number, the solution to the above two expressions is:

$$p_t = \left\lceil \frac{t+J_i}{T_i} \right\rceil \tag{4}$$

Similarly, the last activation that verifies the deadline condition, p_D, is the only value of p that simultaneously verifies:

$$-J_i + (p-1)T_i + d_i \leq D \tag{5}$$

and:

$$-J_i + pT_i + d_i > D \tag{6}$$

from which we get:

$$p \leq \frac{J_i + D - d_i}{T_i} + 1 \qquad \text{and} \qquad p < \frac{J_i + D - d_i}{T_i} \tag{7}$$

so, we get the expression:

$$p_D = \left\lfloor \frac{J_i + D - d_i}{T_i} \right\rfloor + 1 \quad (8)$$

Given that the activations that contribute to the worst case are those with $p \le p_t$ and $p \le p_D$, using eqs. (4) and (8) the worst-case contribution of task τ_i to the busy period is:

$$W_i(t, D) = min\left(\left\lceil \frac{t + J_i}{T_i} \right\rceil, \left\lfloor \frac{J_i + D - d_i}{T_i} \right\rfloor + 1 \right)_0 \cdot C_i \quad (9)$$

Given that d_i may be longer than the period, expression (8) may return a negative value, indicating that all activations have their deadlines after D and so, that the contribution is 0. With $(x)_0$ we indicate that if x is negative the result is 0.

Using this expression we can calculate the worst-case response time of task τ_a. Unfortunately, we don't know which instant in the busy period corresponds to the critical instant, but it is easy to see that it can be found at the beginning of the busy period, or at an instant such that the deadline of the analyzed job of τ_a coincides with the deadline of a task τ_i's job. Otherwise the activation of task τ_a could be moved to an earlier time without changing the execution schedule, but making the response time larger. The set of instants, Ψ, at which the deadline of τ_a's job coincides with the deadlines of one of the task jobs in the busy period is:

$$\Psi = \bigcup \{(p-1)T_i - J_i + d_i\} \quad \forall p = 1 \ldots \left\lceil \frac{L + J_i}{T_i} \right\rceil \quad (10)$$

where L corresponds to longest busy period, calculated as:

$$L = \sum_{\forall i} \left\lceil \frac{L + J_i}{T_i} \right\rceil \cdot C_i \quad (11)$$

The equation above is one of many recurrence equations found in response time analysis [1] in which the value to be calculated is in both sides of the equation; of the many solutions, only the one with the minimum positive value is valid. These equations can be easily solved iteratively by starting with a small value of L and using the value obtained from the equation in the next iteration, until a stable solution is found. The equation is guaranteed to have a solution if the utilization of the task set is under 100%. Although the computation time is pseudopolynomial, it is usually short except for utilizations very close to 100%.

Each potential critical instant is obtained by subtracting d_a from each value in Ψ. Checking all the possible critical instants we can find the critical instant that causes the worst-case response time of the task. Given that there may be several activations of τ_a in the busy period, we must analyze them all. If the first activation of τ_a occurs at time A after the beginning of the busy period, the completion time of activation p of τ_a, $w_a^A(p)$, can be calculated by adding the worst-case contribution of all tasks, which is:

$$w_a^A(p) = B_a + pC_a + \sum_{\forall i \neq a} W_i(w_a^A(p), D^A(p)) \quad (12)$$

where $D^A(p)$ is the deadline of activation p, having the first activation of τ_a occurred at A:

$$D^A(p) = A - J_a + (p-1)T_a + d_a \quad (13)$$

We will assume that if tasks synchronize for using shared resources in a mutually exclusive way they will be using a hard real-time synchronization protocol such as the stack-based protocol [14]. Under this assumption, the effects of synchronization on a task under analysis τ_a are bounded by an amount called the blocking term B_a.

The worst-case response time is calculated by subtracting the activation time from the obtained completion time:

$$R^A(p) = w^A(p) - A + J_a - (p-1)T_a \quad (14)$$

For each value of p, we only need to check the values of A within the period, i.e., between 0 and T_a (if A was greater than the period, then we would be analyzing another activation with a different value of p). That is, we only need to check the values of Ψ in the subset:

$$\Psi^* = \{\Psi_x \in \Psi \,|\, (p-1)T_a - J_a + d_a \le \Psi_x < pT_a - J_a + d_a\} \quad (15)$$

For each element of Ψ^*, named Ψ_x, the value to check is $A = \Psi_x - [(p\text{-}1)T_a\text{-}J_a\text{+}d_a]$.

To calculate the worst-case response time of task τ_a we must determine the maximum response times within all the potential critical instants examined:

$$R_a = max(R_a^A(p)) \quad \forall p = 1 \ldots \left\lceil \frac{L - J_a}{T_a} \right\rceil, \forall A \in \Psi^* \quad (16)$$

III. Mutual Exclusion Synchronization

Baker presents in [14] the Stack Resource Protocol (SRP) for bounding priority inversion in real time systems, independently from the scheduling policy used. The method can be applied to fixed priority or EDF schedulers, for instance. A number called the preemption level is assigned to each task, using the priority or importance of each task: the higher the priority, the higher the preemption level. Shared resources are also assigned a preemption level that is the highest of the preemption levels of all the tasks that may use that resource. And a new scheduling rule is imposed: a task can only get active if its preemption level is strictly higher than the preemption levels of the resources currently locked in the system. With this protocol, in a single processor a task can be delayed by lower priority tasks only once, during the duration of one critical section. For the worst case analysis we just pick the longest. Spuri presents in [6] a technique to optimize the calculation of blocking

times for EDF tasks, based on the fact that not all lower priority tasks may have deadlines that cause a preemption effect on the task under analysis. For simplicity, we don't use those results here, but they could be easily incorporated to the analysis.

For synchronization among EDF tasks we will use the SRP with preemption levels inversely proportional to the local deadline D_i of each task [14]. It is also possible to use the difference between the deadline and the jitter of each task D_i-J_i, as suggested by Spuri [6].

In order to calculate the blocking time for EDF tasks we know, from the properties of the SRP protocol, that at most one critical section from each of them can block the task under analysis. Under that protocol, blocking of any particular task is only possible if the critical section that may delay it has already started before the task is activated. If one critical section with the SRP protocol has started, it is not possible that an EDF task that may block the task under analysis starts executing, because according to the SRP rules it does not have enough level to preempt the critical section.

As a consequence of the blocked-at-most-once property the blocking time for an EDF task τ_i is obtained as:

$$B_i = max(CS_{lm}) \quad (\forall l, \forall (m \neq i) | (Lev(CS_{lm}) \geq Lev_i)) \tag{17}$$

where Lev_i is the preemption level of task τ_i, CS_{lm} is the l-th critical section of task τ_m, and $Lev(CS_{lm})$ is the preemption level of the resource associated with that critical section.

IV. Analysis for Tasks with Static Offsets

A. Computational Model

Now we will consider the offset-based model, in which the real-time system is composed of a set of tasks executing in the same processor and grouped into entities that we call *transactions* [8]. Each transaction Γ_i is activated by a periodic sequence of external events with period T_i, and contains a set of m_i tasks. The relative phasings between the different external events are arbitrary. Each task job is activated (released) when a relative time —called the *offset*— elapses after the arrival of the external event (and then adding the release jitter). In this section of the paper we will assume that the offset is static, i.e., it does not change from one activation to the next. We will consider dynamic offsets in Section 5.

Figure 2 shows an example of such a system: the horizontal axis represents time; down-pointing arrows represent the arrival of the external events associated to each transaction, while up-pointing arrows represent the activation times of each task job; and shaded boxes represent task execution.

Each task will be identified with two subscripts: the first one identifies the transaction to which it belongs, and the second one the position that the task occupies within the tasks of its transaction, when they are ordered by increasing offsets. In this

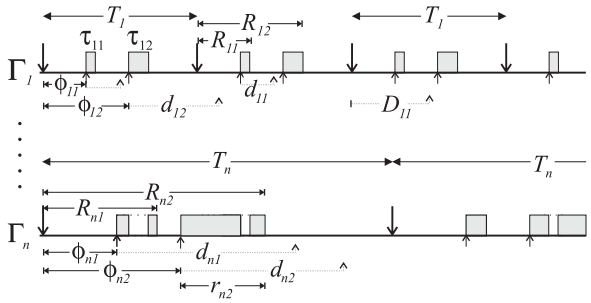

Fig. 2. Computational model of a system composed of transactions with static offsets

way, τ_{ij} will be the j-th task of transaction Γ_i, with an offset of ϕ_{ij} and a worst-case execution time of C_{ij}. In addition, we will allow each task to have a maximum release jitter, which we will call J_{ij}. This means that the activation time of task τ_{ij} may occur at any time between $t_0+\phi_{ij}$ and $t_0+\phi_{ij}+J_{ij}$, where t_0 is the instant at which the external event arrived. Notice that although offsets represent a kind of precedence constraints, in our analysis tasks are activated at a time equal to the arrival of the external event plus the offset and the release jitter, and they execute regardless of whether tasks of the same transaction and smaller offsets have finished or not.

We will allow both the offset ϕ_{ij} and the maximum jitter J_{ij} to be larger than the period of their transaction, T_i. For each task τ_{ij} we can define two kinds of response time: the global response time (also called end-to-end response time), defined as the difference between its completion time and the instant at which the associated external event arrived; and the local response time, measured from the task's offset. We will call R_{ij} the global worst-case response time and r_{ij} the local one. Each task may have a relative deadline d_{ij}, relative to its activation and may have an associated global deadline, D_{ij}, which is relative to the arrival of the external event. We will also allow deadlines to be larger than the periods, and thus at any time there may be several activations of the same task pending.

Tasks are allowed to share resources in a mutually exclusive way, using the mechanism described in Section 3. The blocking term of task τ_{ij} is B_{ij}.

B. Response-Time Analysis

In this subsection we will obtain the worst-case response time of the task under analysis. For building the worst-case scenario for a task τ_{ab} under analysis, we must create a critical instant that leads to the worst-case busy period, with an approach similar to the one described in Section 2. For tasks with offsets, we must take into account that the busy period may not include the simultaneous activation of all tasks, as it was the case when all tasks were independent. The existence of offsets makes it impossible for some sets of tasks to simultaneously become active. The following theorem allows building the busy period in which the worst-case response time of τ_{ab} can be found:

Theorem 2. The worst-case contribution of transaction Γ_i to the response time of a task τ_{ab} is obtained when the first activation of some task τ_{ik} that occurs within the busy period coincides with the beginning of the busy period, after having experienced the maximum possible delay, i.e., the maximum jitter, J_{ik}.

Proof: Similar to the proof of theorem 1.❐

Based on theorem 2, in the analysis of a task τ_{ab} we must study all possible busy periods created by choosing one task in each transaction to coincide with the beginning of the busy period. Given a task τ_{ik} coinciding with the start of the busy period, we will try to find out the contribution of each task τ_{ij} in transaction Γ_i to the worst-case response time. Let us focus on the activation pattern of task τ_{ij}, and let us call its phase relation with the beginning of the busy period, ϕ. This is the time interval between the activation of transaction Γ_i that occurred immediately before the busy period, and the beginning of busy period. Notice that $0 \le \phi < T_i$ because Γ_i is periodic. In order to calculate the worst-case contribution of τ_{ij} to the response time of task τ_{ab} we must categorize each instance of the task into one of the following sets:

- *Set 0*: Activations that occur before the busy period and that cannot occur inside the busy period even with the maximum jitter delay.
- *Set 1*: Activations that occur before or at the beginning of the busy period and that can be delayed by an amount of jitter that causes them to coincide with the beginning of the busy period.
- *Set 2*: Activations that occur inside the busy period.

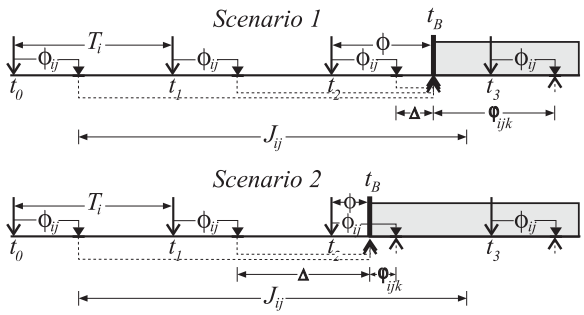

Fig. 3. Scenarios for calculating the contribution of task τ_{ij} to the busy period.

Figure 3 shows two possible scenarios for the alignment of the transaction Γ_i's arrival pattern and the busy period. Scenario 1, in the upper part of the figure, corresponds to the case in which $\phi \ge \phi_{ij}$, and the lower part, Scenario 2, corresponds to $\phi < \phi_{ij}$. Dotted lines represent the actual jitter or delay in the activation time for each instance of the task. In both scenarios, time t_0 corresponds to the first event of Γ_i whose task τ_{ij} may be delayed by jitter until the beginning of the busy period (activations before t_0 would require a delay larger than the maximum jitter to occur at the start of the busy period, t_B). The event that occurs at t_1 may also be delayed by an amount that makes it coincide with the busy period. The activation of τ_{ij} associated with the event that arrived at t_2, can be delayed until the beginning of the busy period in Scenario 1, but not in Scenario 2, because the offset ϕ_{ij} is larger than the relative phase ϕ between the event arrivals and the busy period. For scenario 2, the job associated with the event arriving at t_2 must be included in Set 2.

Once the jobs of task τ_{ij} have been categorized into the three sets above, the calculation of the jitter terms that lead to the worst-case contribution of τ_{ij} to the response time of task τ_{ab} with deadline D is done according to:

Theorem 3. Given a task instance τ_{ab} with deadline D, and a phase relation ϕ between the arrival pattern of transaction Γ_i and the beginning of the busy period, the worst-case contribution of task τ_{ij} to the response time of τ_{ab} occurs when the activations in Set 1 have an amount of jitter such that they all occur at the beginning of the busy period, and when activations in Set 2 have an amount of jitter equal to zero.

Proof: By the definition of a busy period, activations in Set 0 are not involved in it; otherwise, the busy period would have started earlier.

For activations in Set 1, we must delay them with a jitter amount that causes them to occur inside the busy period. But if this delay causes the activation to occur after the beginning of the busy period, it might fall outside the busy period; or its deadline may be delayed to an instant after D, thus making the response time of τ_{ab} smaller. Consequently, to ensure the maximum possible contribution to the busy period, the jitter amount must be such that the activation occurs at the start of the busy period.

For the activations in Set 2, the larger the jitter delay they have, the more probability that the activation occurs outside the busy period or with a deadline after D. Thus, to ensure the worst possible contribution, the jitter amount for these activations must be zero.❐

Under the conditions of Theorem 3, we will now calculate the worst-case contribution of tasks belonging to transaction Γ_i to a busy period of length t and deadline D, when the beginning of the busy period coincides with the activation of one of its tasks τ_{ik}. We will call this contribution $W_{ik}(t,D)$. In the same way as in Section 2, we must consider the activations that occur in the interval $[0,t)$ but with a deadline before or at D. First, we will calculate the number of activations of task τ_{ij} that belong to Set 1, and thus that may accumulate at the beginning of the busy period. We will call this number n_{ij} (in the example, the upper-part scenario had n_{ij}=3 and the lower-part scenario had n_{ij}=2). To calculate n_{ij}, we will define Δ as the difference in time between the time at which the last activation in Set 1 would have occurred if it had no jitter delay, and the start of the busy period. In the example of Figure 3, $\Delta = t_b - t_2$-

ϕ_{ij} for Scenario 1, and $\Delta = t_b - t_1 - \phi_{ij}$ for Scenario 2. It can be seen that:

$$\Delta = \begin{cases} \phi - \phi_{ij} & \text{if} \quad \phi \geq \phi_{ij} \\ T_i + \phi - \phi_{ij} & \text{if} \quad \phi < \phi_{ij} \end{cases} \tag{18}$$

or, equivalently:

$$\Delta = (\phi - \phi_{ij}) \bmod T_i \tag{19}$$

Given that the beginning of the busy period coincides with the activation of task τ_{ik} after having experienced the maximum jitter, phase ϕ is equal to:

$$\phi = (\phi_{ik} + J_{ik}) \bmod T_i \tag{20}$$

so, by the properties of the *mod* function:

$$\Delta = (\phi_{ik} + J_{ik} - \phi_{ij}) \bmod T_i \tag{21}$$

The first activation of τ_{ij} in Set 1 corresponds to the event arriving at t_0, which is the first one whose activation may occur at or after the beginning of the busy period. Therefore, this is the only activation that simultaneously verifies:

$$t_0 + \phi_{ij} + J_{ij} \geq t_B \tag{22}$$

and:

$$t_0 - T_i + \phi_{ij} + J_{ij} < t_B \tag{23}$$

By looking at Figure 3 we can see that:

$$t_B = t_0 + (n_{ij} - 1)T_i + \phi_{ij} + \Delta \tag{24}$$

and replacing it in the two previous expressions we get:

$$\begin{aligned} t_0 + \phi_{ij} + J_{ij} &\geq t_0 + (n_{ij} - 1)T_i + \phi_{ij} + \Delta \\ t_0 - T_i + \phi_{ij} + J_{ij} &< t_0 + (n_{ij} - 1)T_i + \phi_{ij} + \Delta \end{aligned} \tag{25) (25}$$

from which we get:

$$n_{ij} - 1 \leq \frac{J_{ij} - \Delta}{T_i} \qquad and \qquad n_{ij} - 1 > \frac{J_{ij} - \Delta}{T_i} - 1 \tag{26}$$

Given that n_{ij} is an integer number, the solution to the above two expressions is:

$$n_{ij} = \left\lfloor \frac{J_{ij} - \Delta}{T_i} \right\rfloor + 1 \tag{27}$$

In order to determine the effects of activations belonging to Set 2, we need to know the time at which the first of them occurs; the others will occur at periodic intervals after the initial one. We will call φ_{ijk} the time difference between the beginning of the busy period (which coincides with the activation of τ_{ik}) and that first activation in Set 2. Given the definition of Δ we have:

$$\varphi_{ijk} = T_i - \Delta \tag{28}$$

We could have used φ_{ijk} in the equation above to obtain:

$$n_{ij} = \left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor \tag{29}$$

According to Theorem 3, the worst-case contribution of τ_{ij} to a busy period of length t is equivalent to n_{ij} activations at the beginning of the busy period, plus a sequence of periodic activations starting at φ_{ijk} time units after. Without loss of generality, let's set the origin of time at the beginning of the busy period. Then, the worst-case contribution of task τ_{ij} to the response time of τ_{ab} at time t is determined by:

$$\begin{aligned} &n_{ij}C_{ij} + \left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil C_{ij} = \\ &\left(\left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil \right) C_{ij} \end{aligned} \tag{30}$$

where φ_{ijk} can be obtained from (21) and (28) as

$$\varphi_{ijk} = T_i - (\phi_{ik} + J_{ik} - \phi_{ij}) \bmod T_i \tag{31}$$

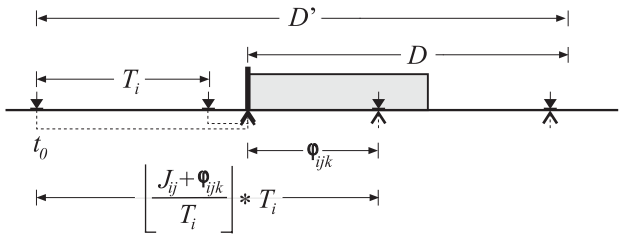

Fig. 4. Activations in busy period with deadline D

Now we must consider the deadline condition, that is, only the activations with deadlines before D can contribute to the response time. In Figure 4 we can see an example of the activations of task τ_{ij} that can contribute to the busy period of length t. There are n_{ij} activations delayed until the beginning of the busy period and an activation that occurs inside the busy period with a phase φ_{ijk}. We can see in Figure 4 that the number of activations in the busy period which have their deadline at or before D is the same as if we consider a periodic task without jitter in a busy period that start at instant t_0 and with a deadline of D'. Given that the interval between t_0 and the first activation in the real busy period is $n_{ij}T_i$, we get:

$$D' = D + \left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor T_i - \varphi_{ijk} \tag{32}$$

and so, the number of activations from instant t_0 that have their deadline at or before D is:

$$\left\lfloor \frac{D' - d_{ij}}{T_i} \right\rfloor + 1 = \left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lfloor \frac{D - \varphi_{ijk} - d_{ij}}{T_i} \right\rfloor + 1 \tag{33}$$

Given that the activations that contribute to the worst case must be at or before the values specified by both conditions (30) and (33), and deriving from (9), the worst-case contribution of task τ_{ij} to a busy period of length t and deadline D when the beginning of the busy period coincides with the activation of τ_{ik} is:

$$W_{ijk}(t, D) = \left(\left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + min\left(\left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil, \left\lfloor \frac{D - \varphi_{ijk} - d_{ij}}{T_i} \right\rfloor + 1 \right) \right)_0 C_{ij} \quad (34)$$

Finally, the total contribution of transaction Γ_i is calculated by adding the contributions of all tasks:

$$W_{ik}(t, D) = \sum W_{ijk}(t, D), \quad \forall j \in \Gamma_i \quad (35)$$

To calculate the worst-case response time of task τ_{ij} we must check all possible busy periods built choosing one task in each transaction. Obviously, the large number of combinations make the analysis intractable and cannot be used. To solve this problem, the following section shows an approximation that gives extremely good results.

C. Upper-Bound Approximation for Worst-Case Analysis

As in the analysis techniques for offsets in fixed priority scheduling [8], we will use the approximate method described in [13] that will let us obtain upper bounds for the global worst-case response times in a system composed of transactions with fixed offsets. Although the technique is not exact, the number of cases that need to be checked has a polynomial dependency on the number of tasks, which makes the method applicable even for relatively large systems. If the response times obtained with this method are smaller than the respective deadlines, the method gives guarantees that all timing requirements will be met.

The main problem with the technique developed in section 4.2 is that we don't know which task τ_{ik} must be used to create the worst-case busy period. This caused us to have to check all possible combinations. We can avoid this problem by obtaining an upper bound to the interference of the tasks of a transaction Γ_i in a busy period of duration t and deadline D, as the maximum of all possible interferences that could have been caused by considering each of the tasks of Γ_i as the one originating the busy period:

$$W_i^*(t, D) = max(W_{ik}(t, D)), \quad \forall k \in \Gamma_i \quad (36)$$

Therefore, using this function in the calculation of the response times, we can make sure that the time obtained is an upper bound for the contribution of the tasks of transaction Γ_i and thus it is not necessary to calculate all the possible combinations for k. By using one function like this for each transaction, we can calculate the global worst-case response time for a particular task by checking only a single case.

In order to introduce less pessimism, we will not use that maximum function for the transaction to which the task under analysis belongs, but we will use the original transaction instead. Consequently, we must repeat the analysis considering each task of transaction Γ_a to be the first to be activated in the busy period, and then use the worst of these results. The number of possibilities is small, equal to the number of tasks in the transaction.

Again, we don't know which instant in the busy period corresponds to the critical instant for the analysis of τ_{ab}, but as we mentioned in Section 2, the critical instant can be found at the beginning of the busy period or at an instant such that the deadline of τ_{ab} coincides with the deadline of another task of another transaction. For convenience, we will number the jobs of tasks using the letter p, with consecutive numbers ordered according to the activation time that they would have had if they had no jitter. In addition, we will assign the value p=1 to the activation of τ_{ij} that occurs in the interval $(0,T_i]$. This means that the activation that occurred in the interval $(T_i,2T_i]$ gets the value p=2, etc. Similarly, the activation that would have occurred in the interval $(-T_i,0]$ but that was delayed to the beginning of the busy period corresponds to p=0, the one in $(-2T_i,-T_i]$ to p=-1, etc. Notice that activations that occur after the beginning of the busy period are numbered with positive numbers, while previous activations have values of $p \leq 0$.

In this schema, the deadline of activation p of task τ_{ij} is equal to $\varphi_{ijk} + (p\text{-}1)T_i + d_{ij}$. Accordingly, the first activation of a task τ_{ij} in the busy period built with τ_{ik} corresponds to index:

$$p_{0,\,ijk} = -\left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + 1 \quad (37)$$

and the last one to:

$$p_{L,\,ijk} = \left\lceil \frac{L - \varphi_{ijk}}{T_i} \right\rceil \quad (38)$$

where L is the length of the busy period (or an upper bound, as obtained in (11)). So, the set of values to check in the analysis, Ψ, can be obtained from:

$$\Psi = \bigcup \{ \varphi_{ijk} + (p-1)T_i + d_{ij} \}$$
$$\forall p = p_{0,\,ijk} \cdots \left\lceil \frac{L - \varphi_{ijk}}{T_i} \right\rceil, \; \forall j, k \in \Gamma_i \quad (39)$$

Given that there may be several activations of τ_{ab} in the busy period, we must analyze them all. In the busy period created with τ_{ac} the activations to analyze are those from $p = p_{0,abc}$ to $p = p_{L,abc}$. If the first activation of τ_{ab} occurs at a time instant A after the beginning of busy period, the completion

time of activation p, $w^A_{abc}(p)$, can be calculated by adding the worst-case contribution of all transactions:

$$w^A_{abc}(p) = B_{ab} + (p - p_{0,abc} + 1)C_{ab} + W^-_{ac}(w^A_{abc}(p), D^A_{abc}(p)) + \sum_{\forall i \neq a} W_i(w^A_{abc}(p), D^A_{abc}(p)) \quad (40)$$

where W^-_{ac} is the result of (35) for transaction Γ_a without considering contribution of τ_{ab}, and $D^A_{abc}(p)$ is the deadline of activation p when the first one occurs at instant A:

$$D^A_{abc}(p) = A + \varphi_{abc} + (p-1)T_a + d_{ab} \quad (41)$$

The global worst-case response time is obtained by subtracting from the obtained completion time the arrival instant of the external event:

$$R^A_{abc}(p) = w^A_{abc}(p) - A - \varphi_{abc} - (p-1)T_a + \phi_{ab} \quad (42)$$

For each p we only need to check the values of A between 0 and T_a (if A was greater, then we would be analyzing an activation with another value of p) in such a way that its deadline coincides with the deadline of another task. That is, we only need to check the values of Ψ in the subset:

$$\Psi^* = \{\Psi_x \in \Psi \mid \varphi_{abc} + (p-1)T_a + d_{ab} \leq \Psi_x < \varphi_{abc} + pT_a + d_{ab}\} \quad (43)$$

For each element of Ψ^*, named Ψ_x, the value to check is $A = \Psi_x - [\varphi_{ijk}+(p\text{-}1)T_a+d_{ab}]$.

The busy period can be calculated by considering the contributions without the deadline limitation, i.e.,

$$L_{abc} = W_{ac}(L_{abc}, \infty) + \sum_{\forall i \neq a} W_i^*(L_{abc}, \infty) \quad (44)$$

To calculate the global worst-case response time of task τ_{ab} we must determine the maximum amount among all the potential critical instants examined:

$$R_{ab} = max(R^A_{abc}(p)) \quad \forall p = p_{0,ijk} \cdots \left\lceil \frac{L - \varphi_{ijk}}{T_i} \right\rceil, \forall c \in \Gamma_a, \forall A \in \Psi^* \quad (45)$$

Note that we can calculate local response times in the same way, but without considering the phase of the task, ϕ_{ab} in equation (42). The system will be schedulable if all worst-case response times (local and global) are lower than the respective deadlines (local and global).

V. Analysis for Tasks with Dynamic Offsets

In this section we will extend the analysis to include the case in which the system has tasks with dynamic offsets. As in the case with static offsets, the system is composed of a set of transactions that execute in the same processor. Each transaction Γ_i has a period of T_i and contains a set of m_i tasks with activation offset Φ_{ij}, worst-case execution time C_{ij}, and maximum jitter J_{ij}. However, in this case task offsets are allowed to vary dynamically, from one activation to the next, within a minimum and a maximum value: $\Phi_{ij} \in [\Phi_{ij,min}, \Phi_{ij,max}]$. In [8] we solved this problem by using the analysis technique for static offsets with an offset equal to $\Phi_{ij,min}$ and a release jitter of $J_{ij}+\Phi_{ij,max}-\Phi_{ij,min}$.

In [8] we saw that the analysis for tasks with dynamic offsets is useful in systems in which tasks suspend themselves and also in distributed and multiprocessor systems. We can model tasks in such systems scheduled under EDF exactly in the same way as we did for fixed priorities. In this kind of system the task offsets and jitter terms depend on the response times of previous tasks in the transaction, which themselves depend on the offsets and jitter terms. For example, in a distributed system with end-to-end deadlines an external event causes a response composed of a sequence of tasks, in which each task activates the next one through the transmission of a message when it finishes. This system can be modeled with tasks having offsets and jitters: each task would have an offset term equal to the earliest possible activation time in the sequence (i.e., the best-case response time of the preceding task) and a jitter term equal to the difference between the worst- and best-case response times of the preceding task.

Using an iterative algorithm, similar to the one used for the holistic analysis, and using adequate initial values for offsets and jitter terms [8] we can calculate the response times. From these we can obtain new jitter terms that we can use to estimate new response times. This algorithm converges to stable values with which we can obtain tight upper bounds for the worst-case response times.

VI. Comparison with existing techniques

We have compared the results of the analysis for tasks with dynamic offsets using the upper-bound approximation presented in Section 4.3 with the results obtained using the current analysis technique for distributed systems by Spuri, which models the system as a set of independent tasks with their release jitter inherited from the previous tasks in the transaction [7]. For this purpose, we have conducted extensive simulations with different task sets whose execution times and periods were generated randomly. Deadlines were assigned based on periods. Local deadlines are assigned by dividing the end-to-end deadline among all the tasks. The results of some of these simulations are shown in this section.

The first set of graphs (Figure 5 to Figure 7) compares the response times obtained using Spuri's technique for independent tasks, R_{indep}, with the response times obtained using our offset-based algorithm, $R_{Offsets}$. In these figures, we show the average ratio $R_{indep}/R_{Offsets}$ obtained for five simulated tasks sets for each point in the graph. The X axis represents processor utilization. Each figure presents the results for three different ratios of the maximum transaction period over the

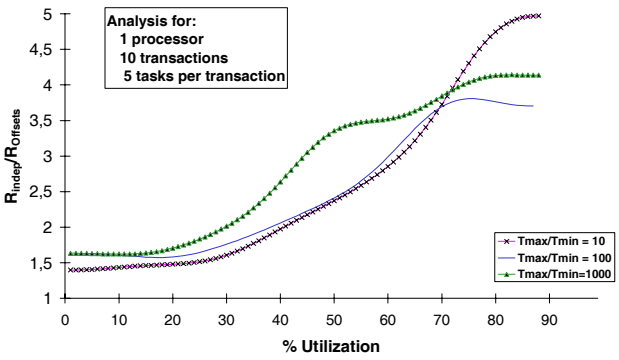

Fig. 5. $R_{indep}/R_{offsets}$, 1 processor, best=0

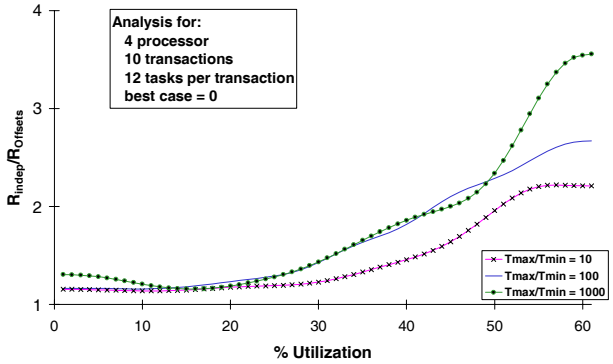

Fig. 6. $R_{indep}/R_{offsets}$, 4 processors, best=0

minimum transaction period, T_{max}/T_{min}. Figure 5 shows the results for a set of 10 transactions with 5 tasks per transaction, in one processor, for the case in which the best-case response times are considered negligible, and thus the task offsets are all zero. It can be seen that for normal utilization levels of around 60%, the response times with independent tasks are roughly between 2.7 and 3.5 times larger than in the analysis with dynamic offsets.

Figure 6 shows the results for a similar case, but running on four processors. We can see that as the number of tasks of the same transaction that are in the same processor diminishes, the benefits of the offset-based algorithm also diminish. However, these benefits are still significant, with response times between 2.2 and 3.4 times better for 60% utilization. Figure 7 shows the results for the same case as Figure 6, except that the best-case response time of each task is considered equal to the sum of the execution times of itself and all its predecessor tasks in the same transaction. We can see that, in this case, the results are significantly better, with response times between 3.5 and 4.75 times better than in the analysis with independent tasks, for a utilization of 60%.

The second set of graphs (Figure 8 and Figure 9) compare the maximum schedulable utilization that can be obtained for a given task set using the analysis for independent tasks, the offset-based algorithm with zero best-case response times, and the

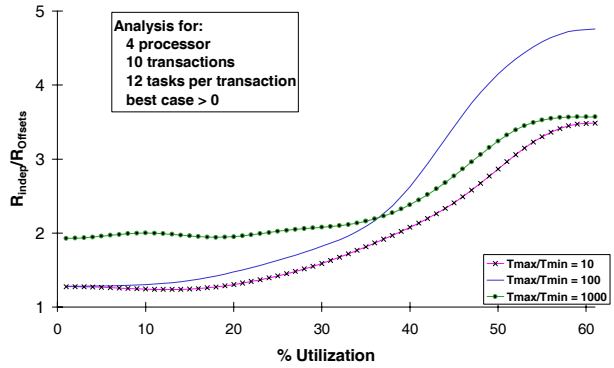

Fig. 7. $R_{indep}/R_{offsets}$, 4 processors, best>0

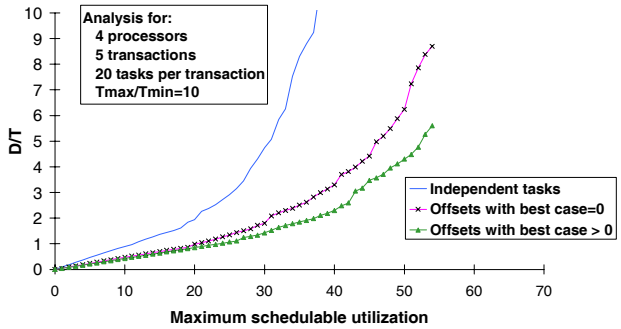

Fig. 8. Maximum scheduling utilization, 100 tasks

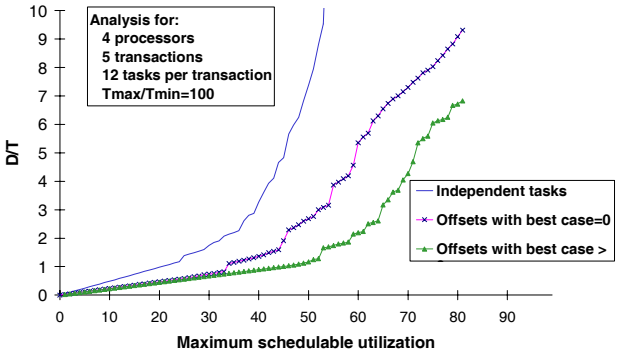

Fig. 9. Maximum scheduling utilization, 60 tasks

offset-based algorithm with best-case response times equal to the task execution times. The maximum schedulable utilization is obtained by analyzing a system with low utilization and then increasing its utilization until the system no longer meets its deadlines. The maximum schedulable utilization is taken for the last of the task sets for which the deadlines were met. The simulations have been done for different ratios of deadlines over periods, D_i/T_i. Figure 8 shows the results for the simulation of a system with 4 processors, 5 transactions and 20 tasks per transaction, with T_{max}/T_{min}=10. Figure 9 shows the results

for a similar task system, but with 12 tasks per transaction instead of 20 and with T_{max}/T_{min}=100. We can see that from values of D_i/T_i=2 and higher, we can get an increase of around 25% more schedulable utilization in the case of 12 tasks, and 16% more in the case of 20 tasks. It is also worth mentioning that for systems with several processors the results are better if we consider best-case response times larger than zero, although it is still possible to get benefits from our new analysis if we consider the best execution times (and thus the offsets) equal to zero.

The execution time of the offset-based analysis techniques for EDF scheduling is significantly higher than for fixed-priority scheduling, because the number of cases to analyze is higher, due to the fact that all the potential critical instants in the busy period need to be checked. In fixed priority systems there is only one critical instant per busy period. In EDF, the amount of cases to check is polynomial, equal to the number of deadlines in Ψ, which is Σm_i^2 , where m_i is the number of tasks in transaction Γ_i. However, since this is an off-line analysis technique it is usually not a problem to spend more time in it.

VII. Conclusions

In this paper we have presented a modification of the analysis of fixed-priority tasks with offsets, to support EDF scheduling. This modification allows analyzing distributed or multiprocessor systems with significantly better results than using the previous holistic analysis for EDF. In many simulation experiments we obtained response times that were between 250% and 500% better than with the holistic analysis. The modification is also useful for analyzing other effects, such as task suspension.

With the technique obtained we have a complete set of tools for analyzing real-time systems with nodes that are scheduled either under EDF or fixed priorities, using the same family of offset-based analysis methods.

References

[1] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, *A Practitioner's Handbook for Real-Time Systems Analysis*. Kluwer Academic Pub., 1993.

[2] J.P. Lehoczky, “Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines". *IEEE Real-Time Systems Symposium*, 1990.

[3] C.L. Liu, and J.W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment”. *Journal of the ACM*, 20 (1), 1973, pp 46-61.

[4] M. González Harbour, M.H. Klein, and J.P. Lehoczky. “Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority”. Proceedings of the *IEEE Real-Time Systems Symposium*, December 1991, pp. 116-128.

[5] S. Baruah, D. Chen, S. Gorinsky S and A. Mok." Generalized multiframe tasks". *Real-Time Systems* 17 (1), pp. 5-22.

[6] M. Spuri. "Analysis of Deadline Scheduled Real-Time Systems". RR-2772, INRIA, France, 1996.

[7] M. Spuri. "Holistic Analysis of Deadline Scheduled Real-Time Distributed Systems". RR-2873, INRIA, France, 1996.

[8] J.C. Palencia and M. González Harbour “Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proceedings of the 19th *IEEE Real-Time Systems Symposium*, 1998.

[9] J.C. Palencia Gutiérrez, J. J. Gutiérrez García, and M. González Harbour, “Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard Real-Time Systems”. 10th *Euromicro Workshop on Real-Time Systems*, Berlin, Germany, June 1998.

[10] L. Sha, R. Rajkumar, and J.P. Lehoczky. “Priority Inheritance Protocols: An approach to Real-Time Synchronization”. *IEEE Trans. on Computers*, Sept. 1990.

[11] J. Sun and J. Liu, “Bounding the end-to-End Response Time in Multiprocessor Real-Time Systems”, Proceedings of the Third *Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, CA, 1995.

[12] K. Tindell, and J. Clark, “Holistic Schedulability Analysis for Distributed Hard Real-Time Systems”. *Microprocessing & Microprogramming*, Vol. 50, Nos.2-3, pp. 117-134, April 1994.

[13] K. Tindell, “Adding Time-Offsets to Schedulability Analysis”, Technical Report YCS 221, Dept. of Computer Science, University of York, England, January 1994.

[14] T.P. Baker, “Stack-Based Scheduling of Realtime Processes”, *Journal of Real-Time Systems*, Vol. 3, Issue 1, March 1991.