Feasibility Analysis of Non-Concrete Real-Time Transactions With EDF Assignment priority

Ahmed Rahni, Emmanuel Grolleau, Michael Richard

Feasibility Analysis of Non-Concrete Real-Time Transactions With EDF Assignment priority

Ahmed Rahni, Emmanuel Grolleau and Michael Richard
LISI/ENSMA
Téléport 2, 1 Av. Clément Ader
BP 40109, 86961 Futuroscope Chasseneuil Cedex
{rahnia,grolleau,richardm}@ensma.fr

Abstract

We present a density based feasibility analysis of tasks with offset (non-concrete transactions) scheduled by preemptive EDF, on a uniprocessor system. Our method extends the analysis technique proposed for sporadic tasks, in [4, 2], that allows a global schedulability analysis of a system. We will show that, for non-concrete transactions, the naive extension analysis is intractable due to an important number of busy periods to consider. To avoid this problem we propose a pseudo-polynomial analysis technique that gives a necessary and sufficient condition of schedulability. This technique is based on the demand bound function. Finally, We provide an efficient implementation, for the presented analysis method, that speeds up the algorithm.

Keywords: Feasibility analysis, Real-Time transactions, EDF, tasks with offsets

1 Introduction

The temporal validation process, in the context of real time systems, is required to guarantee a priori that all the temporal constraints are met (all the tasks complete before their deadlines). The scheduling algorithm allocates the processor to one of the concurrent real-time tasks of a system, the generated execution sequence is called a schedule. A schedule is feasible if all the tasks meet their deadlines. A task system S is feasible with the algorithm A if A generates a feasible schedule for S. We call a schedulability test an algorithm that, given a task set and a scheduling algorithm, returns a negative answer if the scheduling algorithm can generate a non feasible schedule.

A scheduling algorithm A is optimal for a class of scheduling algorithm if for any scheduling algorithm B in the same class as A, if a task system is feasible with B, then it is feasible with A. Since the EDF scheduling is optimal [7], in uniprocessor context for independent task system, then the feasibility test under EDF can be reduced to the schedulability test with EDF.

In the literature, several feasibility tests are proposed for the systems modeled as the well known independent periodic tasks of Liu and Layland [11]. The schedulability conditions obtained with this model are based on the concept of busy period. This is an interval of time during which the processor never goes idle. The first deadline miss (if any) must occur in the longest busy period that is initiated by the simultaneous activation of all the tasks of the system. Unfortunately, theses conditions are too pessimistic for certain kinds of application where the simultaneous activation of some tasks, may not be possible, like tasks with offset [18], multiframe (MF) and generalized multiframe (GMF) tasks [14, 3].

As an example, we present the model of the serial transactions [19], where a data acquisition process is activated by an external event, the acquisition tasks are usually short, because they only have to bufferize the packets until the whole frame is built, while the treatment task is longer since it has to deal with the full frame. The figure 1 presents a temperature acquisition task (with an execution time of 2 units and deadline of 5 time units) activated at the arrival of an external event, this task is followed by a pressure acquisition task (with an execution requiring of 2 time units and deadline 5 time units) activated 5 time units (offset) after the arrival of the temperature frame. The two acquisition tasks are followed by a treatment task that is activated 10 time units after the arrival of the first event.

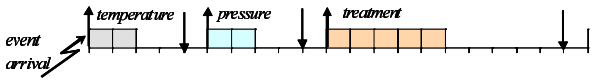

Figure 1. Example of a serial transaction.

It is clear that this task set is feasible because the three tasks are never activated at the same time. Nevertheless, if all the tasks are activated simultaneously (feasibility condition of L&L) then the task set is not feasible, because of the important overestimation of the processor requirement that can be really imposed by the tasks. We note that the instant at which the external event arrives is not known, therefore the activation time of the first acquisition task of the transaction is not known. Thus the transaction is non concrete.

The tasks with offset model (transactions) has been proposed, by Tindell [18], in order to take into account offset relations be-

tween the tasks of a system. Transactions are non-concrete (the transaction activation times are not known a priori). The feasibility problem of transactions has been extensively studied in the context of fixed priority [8, 13] and with dynamic priority [9]. All the existing feasibility test, for transactions, are based on the response time analysis (RTA) [1]. This analysis consists in computing the worst-case response time (WCRT) of each task in the system. If the WCRT of each task is lower or equal than its deadline then it is feasible. Generally, the RTA analysis is based on the study of the critical instant (instant that leads to the worst-case response time of a task) and the busy period. Tindell [18] proved that the critical instant (start of the busy period also) of a task under analysis, is a particular instant when it is activated at the same time as one task of higher priority, after being delayed by a maximum jitter value, in each transaction. Thus several different scenarios are possible, for a transaction set, by combining the tasks of all the transactions. So, the main problem of the exact RTA, is to check the possible scenarios (candidate critical instants), leading to an exponential complexity. Only a sufficient feasibility test with a pseudo-polynomial complexity has been proposed in both the fixed priority context [13, 16] and the dynamic priority context [9].

In this paper, we propose a necessary and sufficient test of the global feasibility of transactions set scheduled by EDF, with a pseudo-polynomial complexity. We use the demand function that has been previously used in feasibility analysis of hard-real-time systems scheduled under EDF priorities [4, 2]. This analysis technique returns an answer concerning the global feasibility of a system without focusing on every task. It consists in proving that for a given interval t, the processor demand of a system, with both activated and deadline dates within t, is always lower than or equal to the interval length t. We note that this technique has a complexity lower than the one of RTA technique. We will show that the naive extension of this technique for transactions is intractable due to the number of considered scenarios (busy periods). We propose an analysis technique based on the maximum demand function (demand bound function $dbf_i(t)$) that computes the maximum cumulative demand of a transaction Γ_i within any interval time of length t; this method provides the same result as the naive extension method with a pseudo-polynomial run time complexity. Finally, we provide an efficient implementation, that allows to speed up the proposed analysis methods, by using the periodicity of both the demand and demand bound functions. We are representing statically the processor demand of transactions in tables, and using it to deduce the processor demand of a transaction for any given busy period of length t.

Section 2 presents the feasibility analysis of sporadic tasks. In section 3, we present the transaction model, we give a definition of the demand function, then we propose a different feasibility analysis method, for transactions. An efficient implementation with a study of the complexity are presented in section 4.

2 Feasibility analysis of sporadic tasks

In this section we present the fundamental results concerning uniprocessor feasibility analysis, for sporadic task systems scheduled under EDF priorities, based on the processor demand criterion [5, 6]. A system S is a set of N sporadic tasks $(\tau_1, \tau_2, ..., \tau_N)$. Each task τ_i is defined by three parameters, a worst-case execution time C_i, a minimal time interval T_i separating two successive activations of τ_i (called jobs or instances) and each instance of τ_i must be executed within a deadline D_i, relative to its activation time. Note that in this model the first activation date of a task is unknown: tasks are said non-concrete. The task set is scheduled under a preemptive EDF scheduler, that is, if some tasks are ready to execute, the scheduler will run the task with the earliest deadline, relative to the current time. A sporadic tasks set is feasible if the corresponding synchronous task set is feasible [4]. i.e. the tasks are simultaneously activated at $t_0 = 0$ and their activations are strictly periodic. Thus when analysing the schedulability of a sporadic task system, we will refer as the underlying periodic task system. The analysis technique is based on the processor demand function [5, 6], which is the amount of time demanded by the tasks in the interval $[t1, t2)$ that the processor must execute to ensure that no task misses its deadline. This function is defined as : $df(t_1, t_2) = \sum_{i=1}^{N} n_i(t_1, t_2) * C_i$, Where n_i is the number of jobs of task τ_i with activation time greater than or equal to t_1 and deadline less then or equal to t_2. A necessary condition for feasibility is: the amount of time demanded by the task set in any interval must never be larger than the length of the interval. $\forall 0 \leq t_1 < t_2 : df(t_1, t_2) \leq t_2 - t_1$.

It has been also showed that, for feasibility analysis, it is sufficient to check the values of $df(t_1, t_2)$ for all the instants t_1 that correspond to the activation time of some job. In the same way, it is sufficient to check only instants t_2 that correspond to the absolute deadline of some job [10, 5]. The time period to study is $H = lcm\{T_1, T_2, ...T_N\}$.

It has been proven, in [4], that the first deadline miss, if any, is found in the longest busy period starting from $t_1 = 0$. A busy period is an interval of time where the processor is never idle. Thus, it suffices to check all deadlines from 0 (begin of the busy period) to the first idle time (length of the busy period). The worst case busy period starts at time $t_1 = 0$ when all the tasks are activated simultaneously. Theorem 1 provides the process of feasibility test of synchronous tasks.

Theorem 1 *[15]. A synchronous task set S is feasible with EDF on a single processor if and only if: $\forall L^* \leq L, df(0, L^*) \leq L^*$ where L^* is an absolute deadline and L is the first idle time in the schedule (the length of the first synchronous busy period).*

The previous result is based on the busy period initiated by the simultaneous activation of all the tasks of a system. For transactions, creating the busy period by the simultaneous activation of all the tasks of all the transactions causes an overestimation of the processor demand (tasks of the same transaction can never be activated at the same time) as it has been showed on an example in the introduction section. In this situation, the definition of the busy period, of L&L task model, leads to a pessimistic feasibility test. To avoid this pessimism, in the next section, we show that the theorem 1 can be applied in the case of transactions then we develop the feasibility analysis, using the same technique.

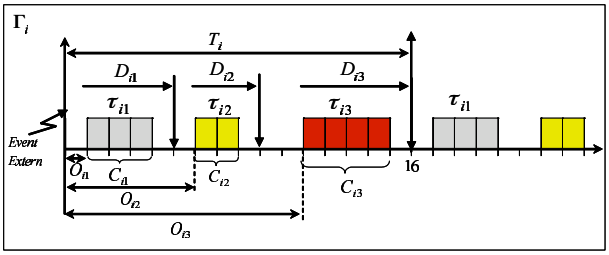

Figure 2. Example of transaction.

3 Feasibility analysis of real-time transaction

3.1 Computational model

A tasks system S is composed of a set of $|S|$ transactions Γ_i, with $1 \leq i \leq |S|$ (where $|S|$ is the number of elements in the set S).

$$\begin{aligned} S &: \{\Gamma_1, \Gamma_2, .., \Gamma_{|S|}\} \\ \Gamma_i &: \{\tau_{i1}, \tau_{i2}, ..., \tau_{i|\Gamma_i|}, T_i\} \\ \tau_{ij} &: < C_{ij}, O_{ij}, D_{ij}, J_{ij} > \end{aligned}$$

Each transaction Γ_i (see Figure 2) consists of a set of $|\Gamma_i|$ tasks τ_{ij} activated at the same period T_i, with $0 < j \leq |\Gamma_i|$. Without loss of generality, we suppose that the tasks are ordered in the set by increasing offset. A task τ_{ij} is defined by : a worst-case execution time (WCET) C_{ij}, an offset O_{ij} related to the activation date of the transaction Γ_i, a relative deadline D_{ij} (related to the offset), a maximum jitter J_{ij} (the activation time of a task τ_{ij} may occur at any time between $t_0 + O_{ij}$ and $t_0 + O_{ij} + J_{ij}$, where t_0 is the activation date of the transaction Γ_i).

The Figure 2 presents an example of a transaction Γ_i composed of three tasks with period $T_i = 16$. Note that each transaction is non-concrete (in fact it's sporadic).

3.2 Feasibility analysis technique

In the context of dynamic priorities scheduling (EDF), for a classical independent tasks, 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 [17]. A candidate critical instant can occur at the beginning of the busy period, or at an instant such that the deadline of the analyzed instance, coincides with the deadline of a task instance of the system. This result has been extended, by Palencia and Harbour [9], for transactions. They showed that for computing the exact WCRT of an analyzed task, one needs to test all the possible busy periods (same as in fixed priorities), that makes the exact analysis intractable (exponential complexity). Then, they proposed a sufficient (approximated) analysis with a pseudo-polynomial time complexity.

In this section, we present the analysis technique based on the demand bound function [4]. This technique allows an exact test of the global schedulability of a system with EDF. Effectively, this method, for a given system scheduled by EDF, returns a boolean answer (feasible or not) concerning its feasibility, but no information on the worst case response time of tasks. So, by using this analysis for transactions scheduled by EDF, we provide an exact schedulability test, in a pseudo-polynomial time complexity.

Baruah et al [3] show that for applying the theorem 1 on a task model, the task independence assumptions [3] must be satisfied. In transactions the runtime behavior of each task is independent of the behavior of other tasks in the system and the tasks are triggered by an external event (arrival date is not a priori known). The transactions are non-concrete and all temporal specifications are made relative to the arrival date of the external event. Thus the transaction model satisfies theses assumptions, therefore we can use the theorem 1 in order to analysis the schedulability of transactions scheduled by EDF. But the problem is to identify the busy period, among all the possible scenarios, in which the feasibility test must be applied. Before addressing this problem, we show how the processor demand can be computed for a given scenario.

The feasibility analysis by demand criteria consist in capturing the demand (interference) of activated tasks that must be finished in an interval of length t (where $t = 0$ at the beginning of the busy period). We give a definition of demand function, in the context of transactions, for a given busy period starting at the simultaneous activation of a candidate task τ_{ic} in each transaction Γ_i.

Definition 1 *(Demand Function). Let Γ_i be a transaction, a busy period is starting by the activation of a task τ_{ic} of Γ_i, and a positive integer number t. The demand function $df_{ic}(t)$ denotes the cumulative execution requirement by jobs of all tasks τ_{ij} of Γ_i that have activation times within any time interval of duration $J_{ij} + t$ and deadlines within time interval of duration t.*

Note that we have obtained the following formulas, that calculate the processor demand, are derived the same way as the derivation of the formulas of interference in [9], the difference is that we use only one variable t, for both activation and deadline dates, for the demand function, while in [9], two variables are used, t for activation dates and d for deadline dates.

In order to compute the worst processor demand of a system in a busy period of any length t, we must to find the worst demand (interference) of each task τ_{ij} of each transaction Γ_i. For each possible busy period, we note τ_{ic} the candidate task in Γ_i initiating the studied busy period. We focus on the activation pattern of τ_{ij} (Figure 3); Φ_{ijc} denote the phasing between τ_{ij}, and the beginning time of the busy period initiated by the activation of the candidate task τ_{ic}; i.e the first instance of a task τ_{ij} (activated within the the busy period) will be activated at Φ_{ijc} time units after the beginning of the busy period, and subsequent activations will occur periodically every T_i. Note that $0 \leq \Phi_{ijc} < T_i$.

$$\Phi_{ijc} = (O_{ij} - (O_{ic} + J_{ic}))\ mod\ T_i \quad (1)$$

In the sequel, a notation x_{ijc} will be related to the task τ_{ij} in a scenario where τ_{ic} is activated at the critical instant.

Since a task τ_{ij} can interfere with multiple instances during a busy period of length t. The instances of τ_{ij} activated in the busy period of t times can be categorized into two sets.

- Set1: Instances activated before or at the beginning of the busy period and that can be delayed by a release jitter so that they coincide with the beginning of the busy period.

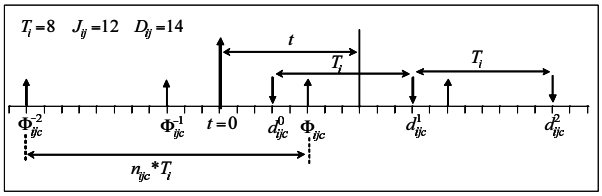

Figure 3. Pattern of activations and deadlines in busy period.

- Set2: Instances activated after the beginning of the busy period.

Based on these two sets and the phasing times, we will be able to calculate the worst-case processor demand of tasks belonging to transaction Γ_i during a busy period of length and deadline t, we will call $df_{ijc}(t)$ the demand function of a task τ_{ij} in a busy period which beginning coincides with the activation of τ_{ic}. For this, we must consider only the instances of set1 and set2 with a deadline before or at t.

The worst processor demand of τ_{ij} within a time interval t is divided into two part. df_{ijc}^{set1} the demand caused by the instances of Set1 with deadline in t, and df_{ijc}^{set2} the demand caused by the instances of Set2 with a deadline before t also. Let us note n_{ijc} the number of instances of τ_{ij} belonging to Set1. in the example of Figure 3 there are $n_{ijc} = 2$ instance that are activated before the beginning of the busy period and delayed enough by jitter to be released in the busy period.

$$n_{ijc} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor \tag{2}$$

From n_{ijc} activations in Set1, we must consider, in the demand function, only the instance activations with deadlines before t . In Figure 3, only the first instance activation (Φ_{ijc}^{-2})) of set1 has a deadline d_{ijc}^{0} that occur before $t = 8$. Thus the number of activations from instant t_0 that have their deadline at or before t is obtained by:

$$\left\lfloor \frac{n_{ijc}T_i + t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \tag{3}$$

therefore, the demand function df_{ijc}^{set1} computing the processor demand of instances belonging in set1 is defined as:

$$df_{ijc}^{set1}(t) = \left(\min \left(n_{ijc}, \left\lfloor \frac{n_{ijc}T_i + t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \right) \right)_0 C_{ij}$$

Where $(x)_0$ is $\max\{x, 0\}$.

With the same principle, we calculate the demand caused by the instances belonging to Set2. We know that Φ_{ijc} is the time at which the first instance of them occurs; the others will occur at periodic intervals after the initial one. We give the function $df_{ijc}^{set2}(t)$:

$$df_{ijc}^{set2}(t) = \left(\min \left(\left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil, \left\lfloor \frac{t - D_{ij} - \Phi_{ijc}}{T_i} \right\rfloor + 1 \right) \right)_0 C_{ij}$$

Note that the precedent equations may return a negative value because the relative deadline may be greater than the period. It is obviously that, for $t \leq d_{ijc}^{0} + (n_{ijc} - 1)T_i$, $df_{ijc}^{set2}(t)$ is null because only the deadlines, of instances of set1, occur in this interval time. And for $t \geq d_{ijc}^{0} + (n_{ijc} - 1)T_i$, $df_{ijc}^{set1}(t) = n_{ijc}C_{ij}$.

$$df_{ijc}(t) = df_{ijc}^{set1}(t) + df_{ijc}^{set2}(t) \tag{4}$$

$df_{ijc}(t)$ is the processor demand caused by τ_{ij} in a busy period of length t, when its beginning coincides with the activation of the task τ_{ic}. Thus the total demand $df_{ic}(t)$, caused by a transaction Γ_i, is given by:

$$df_{ic}(t) = \sum_{\forall j} \left(df_{ijc}^{set1} + df_{ijc}^{set2} \right) \tag{5}$$

Now, we have all the ingredients for checking the feasibility of a system, for a given busy period. For This, one has to check if all deadlines, in the busy period, of each task in the system, is met, using the following test formula:

$$\forall L^* \leq L, \sum_{\forall \Gamma_i} df_{ic}(L^*) \leq L^* \tag{6}$$

Where L denotes the length of the busy period. L is obtained by finding, iteratively, the fix point, using the demand processor as in the context of fixed priority (without regarding the job's deadline).

$$L^{(n+1)} = \sum_{\forall i, \forall j} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{L^{(n)} - \Phi_{ijc}}{T_i} \right\rceil \right) C_{ij} \tag{7}$$

The main problem of feasibility analysis of tasks with offsets is that we dont know which task τ_{ic} of each transaction Γ_i must be considered to create the worst-case busy period. Thus, as a naive solution, we must check the feasibility of a system in all possible busy periods by carrying out all the possible combinations of the tasks in each transaction. We note $C :< c_1, c_2, ..., c_{|S|} >$ a combination of index of the candidate task τ_{ic_i} in each transaction Γ_i initiating a given busy period. Θ is the set of all possible combinations C (with $i \in 1..|S|$ and $c_i \in 1..|\Gamma_i|$). The exact analysis consists in checking the following test:

$$\forall t \leq L, \max_{C \in \Theta} \sum_{\forall \Gamma_i} df_{ic_i}(t) \leq t \tag{8}$$

Obviously, the large number of combinations ($\prod_{i=1}^{|S|} |\Gamma_i|$ busy periods) makes the analysis of every combination intractable. We can solve this problem, by using the demand bound processor criteria that computes an upper bound of the processor demand for a system for any interval time t. This technique provides the same result as the one checking all the possible busy period (Theorem 2).

We define $dbf_i(t)$ the demand bound function : it is the upper bound of the processor demand caused by a transaction Γ_i in a busy period of duration t. It is the maximum of all possible demand that could have been caused by considering each candidate task τ_{ic}, in Γ_i, as the one originating the busy period.

Definition 2 *(Demand Bound Function.) Let Γ_i be a transaction. The demand bound function $dbf_i(t)$ denotes the maximum cumulative execution requirement by jobs of the all task of Γ_i that have both arrival times and deadlines within any time interval of duration t. Formally:*

$$dbf_i(t) = \max_{\forall \tau_{ic} \in \Gamma_i} df_{ic}(t) \tag{9}$$

In order to test the feasibility of a system, one has to check that all the absolute deadlines in the longest busy period are meet, i.e checking for each deadline (time interval t) of each task in the system that the dbf is lower than or equal to t. Thus the test process has a pseudo-polynomial dependency on the number of tasks, which makes the method applicable even for relatively large systems.

Theorem 2 *A transaction system is feasible if and only if $\sum_{\forall \Gamma_i} dbf_i(t) \leq t$ for all positive number t.*

Proof:

The proof consists in showing algebraically the equivalence between the left part of the inequality 8 and the one of theorem 2. For a time interval of duration t, we note $df_{ic_i}(t)$ the processor demand, within t, of Γ_i when τ_{ic_i} initiate the busy period. Assume $\theta =< df_{1c_1}(t), df_{2c_2}(t), ..., df_{nc_{|S|}}(t) >$ denotes the set of interferences of the all transactions of the system for a given busy period initiated by a candidate task τ_{ic_i} in each transaction Γ_i. Note $\Theta = \{\theta : \forall i \in 1..|S|,\ c_i \in 1..|\Gamma_i|\}$ the set of all possible scenarios (busy periods). The value of processor demand function (equation 8) of a system, in the exact test is given by:

$$dbf(t) = \max_{\theta \in \Theta} \left(\sum_{i=1}^{|S|} df_{ic_i}(t) \right)$$

$df_{ic_i}(t)$ is the $i'th$ element of θ. i.e the interference of the $i'th$ transaction (Γ_i) for the given combination θ. Since each processor demand $df_{ic_i}(t)$ of a transaction Γ_i is independent of all other transaction $\Gamma_j \neq \Gamma_i$, then the elements of the all sets θ are independents. Thus we can swap the summation and maximization operations. The formula of demand bound function of the system becomes:

$$dbf(t) = \sum_{i=1}^{|S|} \left(\max_{\theta \in \Theta} df_{ic_i}(t) \right)$$

In each combination θ the $i'th$ element corresponds to the interference of Γ_i. Thus this element is always in the set of interferences $\{df_{ic_i} : c_i \in 1..|\Gamma_i|\}$. For a given i we have

$$\max_{\theta \in \Theta} df_{ic_i}(t) = \max_{c_i = 1..|\Gamma_i|} df_{ic_i}(t)$$

The processor demand becomes:

$$dbf(t) = \sum_{i=1}^{|S|} \left(\max_{c_i = 1..|\Gamma_i|} df_{ic_i}(t) \right)$$

By definition of the demand bound function (equation 9), we can rewrite the formula as following:

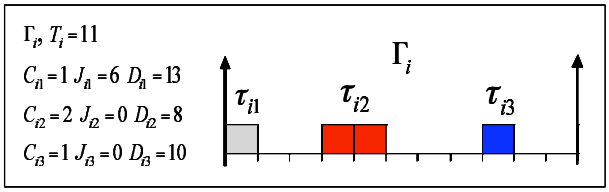

Figure 4. Example of transaction with jitters.

$$dbf(t) = \sum_{i=1}^{|S|} dbf_i(t)$$

So, the exact test of equation 8 is equivalent to the following test (theorem 2)

$$\forall t \leq L, \sum_{\forall \Gamma_i} \left(\max_{c \in 1..|\Gamma_i|} df_{ic}(t) \right) \leq t \tag{10}$$

□

After the result of theorem 2, we conclude that the maximization of the processor demand function on each transaction, does not only give a sufficient condition but a necessary and sufficient schedulability test i.e if the system is not feasible by the dbf function then we are sure that the system is not feasible. This result is interesting compared to [9] where the author focus on RTA of transactions scheduled by EDF. In the case of RTA, a sufficient condition is given.

Applying the result of theorem 1 of baruah [5], the procedure of checking whether a system S is infeasible, consists in determining if there exists an absolute deadline L^*, in the upper busy period of length L, such that the total demand bound, in L^*, is greater than the duration L^*.

$$\exists L^* \leq L : \left(\sum_{\Gamma_i \in S} dbf_i(L^*) \right) > L^* \ ? \tag{11}$$

The length of the upper busy period is computed iteratively using the maximum interference in the context of fixed priority (the deadline is not taken in account)

$$L^{(n+1)} = \sum_{\forall \Gamma_i} W_i(L^{(n)}) \tag{12}$$

$W_i(t)$ denotes the maximum cumulative demand caused by all the job's (activated in t) tasks of a transaction Γ_i, in any interval time t.

$$W_i(t) = \max_{\forall \tau_{ic}} \left\{ \sum_{\forall j} \left(\left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor + \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil \right) C_{ij} \right\} \tag{13}$$

For our example of Figure 4, we calculate the demand of the transaction Γ_i, in a time interval $t = 23$, when each task τ_{i1}, τ_{i2}, or τ_{i3} initiate the busy period. Then the maximum

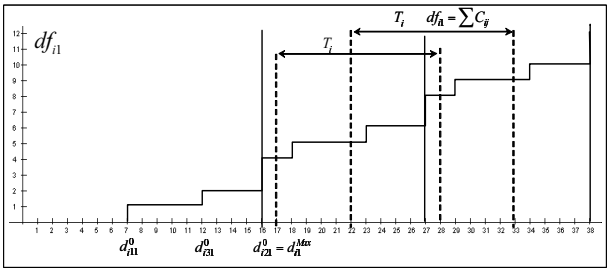

Figure 5. demand function of transaction for a given busy period.

among the demands is considered as a bound on the demand of Γ_i.

$$df^{set1}_{i11}(23) = \left(\min\left(1, \lfloor \tfrac{1*11+23-5-13}{11} \rfloor + 1\right)\right)_0 * 1 = 1$$

$$df^{set2}_{i11}(23) = \left(\min\left(\lceil \tfrac{23-5}{11} \rceil, \lfloor \tfrac{23-5-13}{11} \rfloor + 1\right)\right)_0 * 1 = 1$$

Thus $df_{i11} = df^{set1}_{i11} + df^{set2}_{i11} = 2$. With the same way we calculate the demand caused by τ_{i2} ($df_{i21} = 2$), and the one caused by τ_{i3} ($df_{i31} = 2$). The demand caused by the transaction is $df_{i1} = 2 + 2 + 2 = 6$ (Figure 5).

When τ_{i2} initiates the busy period we have $df_{i2} = 7$, and when τ_{i3} initiates it we have $df_{i3} = 5$. From these values of demands caused by Γ_i the bound demand is the maximum $df_i = 7$.

This method has a pseudo-polynomial time complexity, the demand bound computation, of a transaction Γ_i has a complexity: $O(|\Gamma_i|^2)$ because for each candidate task τ_{ic}, that initiates the busy period, we must calculate the demand caused by each task τ_{ij}. Let us note X the number of deadlines to check, in the longest busy period, then the complexity of the method is $O(X\ |\Gamma_i|^3)$ (assume $|\Gamma_i| = |S|$).

4 Faster feasibility analysis

We see, that when testing the feasibility of a system, we calculate repetitively (according to the number of deadlines to check), for each transaction Γ_i, the $dbf_{ijc}(t)$ for each pair of tasks (τ_{ic}, τ_{ij}). Thus an important amount of computation effort can be avoided by finding a repetitive pattern of dbf_i. Therefore, in this section, we propose a new and efficient implementation algorithm that reduces significantly the time needed by the feasibility analysis method. This technique is based on the same idea as transactions with fixed priorities [12, 13]. The idea is to find a periodic and static pattern of demand bound function of transactions (stored in tables) and to use it in order to compute efficiently the $dbf(t)$ for any t in the different steps of the feasibility test. This method significantly speeds up the feasibility test as showed in the context of fixed priorities [13, 12].

4.1 The periodicity of the demand function

We know that the task activation dates are periodic, this implies that deadline dates are also. In a given busy period initiated by the activation of τ_{ic}, the first deadline of a task τ_{ij} to meet, that we note d^0_{ijc} is the deadline of the first instance of Set1 if it is not empty, else it is the deadline of the first instance of Set2. The first deadline of τ_{ij}'s jobs (activated in the busy period) to meet in the busy period is given by:

$$d^o_{ijc} = \Phi_{ijc} + D_{ij} - n_{ijc}T_i \tag{14}$$

We can express the demand caused by τ_{ij} during a busy period $t \leq d^0_{ijc} + k*T_i$ as $(k+1)*C_{ij}$. We see that during each interval of length T_i after d^0_{ijc} the demand (not cumulative) caused, by τ_{ij}, is always C_{ij} (one activation each T_i times). We note d^{Max}_{ijc} the maximum among the first deadlines of all tasks of Γ_i.

$$d^{Max}_{ic} = \max_{\forall \tau_{ij}} \left\{d^0_{ijc}\right\} \tag{15}$$

Using this report, we deduce that the processor demand of a transaction is periodic after d^{Max}_{ic}, and the processor demand(no cumulative) caused by a transaction Γ_i, during each time interval of length T_i (after d^{Max}_{ic}) equal to $\sum_{\forall \tau_{ij}} C_{ij}$ (no cumulative); Figure 5 show that for $t > d^{Max}_{ic}$ the demand is repetitive (with period = T_i) , this allows us to obtain a static representation of df_{ic} function during any time $t \geq d^{Max}_{ic}$.

In order to represent statically the $dbf_i(t)$ function, that is the maximum of possible demands, of Γ_i, we need to represent statically all the demands df_{ic} for each task τ_{ic}, of Γ_i, during the same time interval (large interval for which the dbf_{ic} are not repetitive). For an easy deduction of df_i during an arbitrary interval t (Theorem 3) we represent the demand functions for each task candidate τic during the interval time $d^0_{max} + T_i$.

$$d^0_{max} = \max_{\forall c, \forall j} \left\{d^0_{ijc}\right\} = \max_{\forall c} \left\{d^{Max}_{ic}\right\} \tag{16}$$

4.2 Static representation

For each critical instant candidate, τ_{ic}, we define a set of points P_{ic}, where each point $P_{ic}[k]$ has an x (representing deadline time) and a y (representing the cumulative processor demand caused by job's tasks of Γ_i with activations and deadlines before x) coordinates, describing how the cumulative processor demand caused with deadline time when the activation of τ_{ic} begins the busy period. The points in P_{ic} correspond to the convex corners of $df_{ic}(t)$ illustrated by dots in Figure 6 . We note Ω_{ic} the ordered and non redundant set of couple deadline dates d (occurred before or at $d^0_{max} + T_i$) and the the corresponding demand C, caused only by the instances of the same deadline d, of all tasks of Γ_i.

$$\Omega_{ic} = \left\{d^0_{ijc} + k*T_i \leq d^0_{max} + T_i,\ \forall j, \forall k = 0, 1...\right\} \tag{17}$$

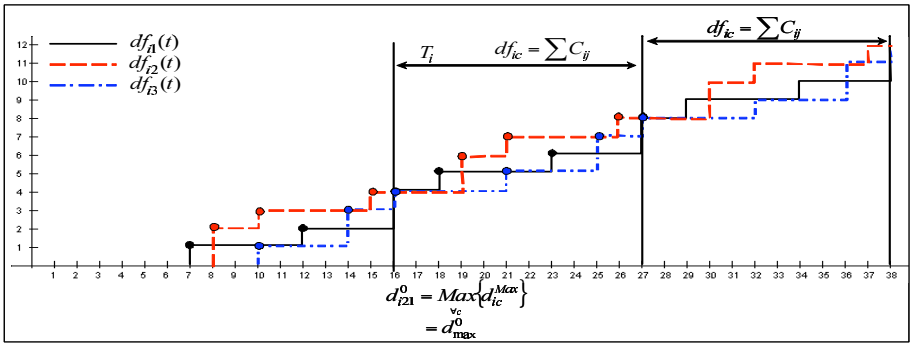

Figure 6. pattern of the demand functions of a transaction for all the busy periods.

The following equations define the array P_{ic}:

$$\begin{aligned} P_{ic}[1].x &= \Omega_{ic}(1).d \\ P_{ic}[1].y &= \Omega_{ic}(1).c \\ &\ldots \\ P_{ic}[k+1].x &= \Omega_{ic}(k+1).d \\ P_{ic}[k+1].y &= P_{ic}[k].y + \Omega_{ic}(k+1).c \end{aligned} \tag{18}$$

For our example of Figure 6, the P_{ic} table that stores the coordinate of the convex point of each curve corresponding to df_{ic} are given as:

$P_{i1} =< (7,1),(12,2),(16,4),(18,5),(23,6),(27,8) >$
$P_{i2} =< (8,2),(10,3),(1,4),(19,6),(21,7),(26,8) >$
$P_{i3} =< (10,1),(14,3),(16,4),(21,5),(25,7),(27,8) >$

Once the informations on all df_{ic}-functions are stored in P_{ic}-tables, in order to store the information about the bound demand dbf_i function, we define the table of points, P_i, as the union of all P_{ic}-s

$$P_i = \bigcup_{\forall \tau_{ic}} P_{ic} \tag{19}$$

In order to determine the points in P_i corresponding to the convex corners of $dbf_i(t)$ (Figure 7), describing the demand bound function, we define a subsumes relation: A point $P_i[a]$ subsumes a point $P_i[b]$ (denoted $P_i[a] \succ P_i[b]$) if the presence of $P_i[a]$ implies that $P_i[b]$ is not a convex corner. Graphically, the set P_i is illustrated by the dots in Figure 7.

$$\begin{aligned} &P_i[a] \succ P_i[b] \;\; if \; and \; only \; if \\ &(P_i[a].x \leq P_i[b].x \;\wedge\; P_i[a].y \geq P_i[b].y) \end{aligned} \tag{20}$$

Using the subsume relation, we will remove from P_i all subsumed points (that will not be used to represent the dbf_i function i.e the convex corner of curve that represents dbf_i):

$$Remove \;\; P_i[b] \;\; from \;\; P_i \;\; if \;\; \exists a \neq b : P_i[a] \succ P_i[b] \tag{21}$$

For our transaction of Figure 6. We keep in $P_i = p_{i1} \cup p_{i2} \cup p_{i3}$ only the points representing the maximum dbf_i.

$P_i =< (7.1),(8.2),(10.3),(15.4),(18.5),(19.6),(21.7),(26.8) >$

At this step, the dbf_i is stored in P_i (convex corners of dbf_i function illustrated by the dots in Figure 7). The following theorem tells how we can deduce the bound demand of Γ_i for any time t, using P_i.

Theorem 3 *During an interval time $t = t' + m * T_i$, where $d^0_{max} \leq t' < d^0_{max} + T_i$, the demand bound caused by a transaction Γ_i is: $dbf_i(t) = dbf_i(t') + m * \sum_{\forall \tau_{ij}} C_{ij}$.*

proof:

We can prove this deduction easily, by algebraic equivalence: Let $t = d^0_{max} + m * T_i + t^*$, where $t^* < T_i$, be a duration of a busy period. We note $dbf_i(a,b) = dbf_i(b) - dbf_i(a)$ the demand caused only during the interval that starts at date a, in a busy period, and finishes at date b. Since, after a time $t > d^0_{max}$, the demand caused by a transaction is periodic with period T_i and during each time interval of duration T_i an amount of $\sum_{\forall \tau_{ij}} C_{ij}$ time requirement is caused (Figure 6), then the demand caused between $d^0_{max} + kT_i$ and $d^0_{max} + kT_i + t^*$ is equivalent to the on caused between d^0_{max} and $d^0_{max} + t^*$.

$$dbf_i(d^0_{max} + kT_i, d^0_{max} + kT_i + t^*) = dbf_i(d^0_{max}, d^0_{max} + t^*)$$

Thus

$$\begin{aligned} dbf_i(t) &= dbf_i(0, d^0_{max}) + m * \sum_{\forall \tau_{ij}} C_{ij} + \\ &\quad dbf_i(d^0_{max}, d^0_{max} + t^*) \\ dbf_i(t) &= dbf_i(d^0_{max}) + m * \sum_{\forall \tau_{ij}} C_{ij} + \\ &\quad dbf_i(d^0_{max} + t^*) - dbf_i(d^0_{max}) \\ dbf_i(t) &= m * \sum_{\forall \tau_{ij}} C_{ij} + dbf_i(d^0_{max} + t^*) \\ dbf_i(t) &= m * \sum_{\forall \tau_{ij}} C_{ij} + dbf_i(t') \end{aligned}$$

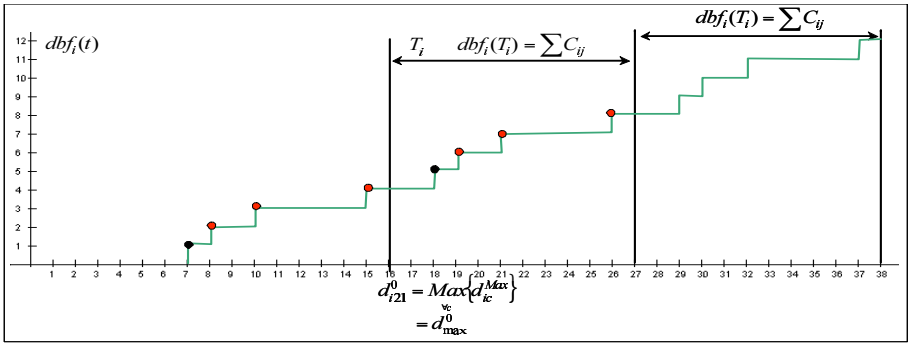

Figure 7. Demand bound function of transaction.

with

$$t^{'} = d^0_{max} + t^* \ \ and \ \ d^0_{max} \leq t^{'} < d^0_{max} + T_i$$

□

Using P_i, the maximum demand caused by a transaction Γ_i, during a interval time $t \leq d^0_{max} + T_i$, is obtained by a fast lookup function, as follow:

$$\begin{aligned} dbf_i(t) &= P_i[n].y \\ n &= \max\{m : P_i[m].x \leq t\} \end{aligned} \tag{22}$$

In order to validate a system S, the set ψ of deadline dates to check is reduced to the dates at which the $dbf_{i,\forall i}$ changes value, Thus we need to check that the demand bound caused by a system is always lower or equal than the deadline belonging to ψ.

$$\psi = \bigcup_{\forall \Gamma_i \in S} \psi_i \tag{23}$$

$$\begin{aligned} \psi_i = \{d = P_i[k].x \ / \ k = 1..|P_i|\} \cup \\ \{\ d = P_i[k].x + m * T_i \ / P_i[k].x > d^0_{max} \ \wedge \ m = 1..\} \end{aligned}$$

For the same example of Figure 4,we compute the bound demand of Γ_i, for $t = 32$. We use the table P_i for calculate the demand bound for any $t < d^0_{max} + T_i = 27$.

We have $t = 21 + 1 * T_i = 21 + 11$ ($m = 1$ and $d^0_{max} = 16 \leq 21 < d^0_{max} + T_i = 27$. Thus $dbf_i(32) = dbf_i(21) + 1 * \sum_{\forall \tau_{ij}} C_{ij} = 7 + 4 = 11$ (Figure 7). From the table P_i, we deduce that $dbf_i(21) = P_i[7].y = 7$.

4.3 Time complexity

In the context of sporadic tasks scheduled under dynamic (EDF) priority, [4, 2] proved that for $U < 1$ the length of the busy period is bounded by $\ell = \left(\frac{U}{1-U} \max_{i=1}^{N}\{T_i - D_i\}\right)$ and the time complexity of the feasibility analysis is $O(N.\ell)$, with N number of tasks and $U = \sum_{i=1}^{N}(C_i/T_i)$ the utilization factor of the system.

The feasibility analysis of transactions is divided into two phases. First, the static representation of the demand bound function, of each transaction Γ_i, during a time interval $t \leq d^0_{max} + T_i$; This operation, for one transaction Γ_i, has a complexity $O\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right)|\Gamma_i|^2 . \log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right)|\Gamma_i|\right)\right)$. The second phase determines the feasibility of the system, using the lookup tables obtained during the first phase. Searching a value in a table P_i needs a time in $O\left(\log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right)|\Gamma_i|\right)\right)$. We note X the number of deadlines, that we have to check. Thus the overall complexity of the second phase is $O\left(X \log\left(\left(1 + \frac{d^0_{max} - d^0_{min}}{T_i}\right)|\Gamma_i|\right)\right)$.

In the case, when $D_{ij} < T_i$ the complexity is reduced to $O\left(|\Gamma_i|^2 . \log(|\Gamma_i|)\right)$ for the static representation, and for the feasibility analysis it is $O(X \log|\Gamma_i|)$. Thus the overall complexity is in $O\left(|\Gamma_i|^2 . \log(|\Gamma_i|) + X \log|\Gamma_i|\right)$

For transactions scheduled under EDF, a response time analysis (RTA) has been proposed by Palencia and Harbour [9]. The sufficient analysis proposed consists in computing a bound for a response time for each task in a system, using the maximum (approximate) interference function. Giving a task under analysis τ_{ua}, Let X be the number of deadlines, of the tasks of a system, in an upper busy period. Y is the number of iterations for computing a response time (finding a fixed point) of an instance of τ_{ua} for a given critical instant, and Z the number of instances of τ_{ua} activated in a busy period according to the critical instant considered. The maximum interference calculation, of a transaction, during an interval time t, has a complexity $O(|\Gamma_i|^2)$. The RTA consists in computing the response time of the all instances, activated in the busy period, of the task under analysis, for each critical instant in all possible busy periods. Thus the complexity of the RTA algorithm, for τ_{ua} is $O(X * Y * Z * |S| * |\Gamma_i|^2)$. The analysis is performed for all the tasks of a system in $O(X * Y * Z * |S|^2 * |\Gamma_i|^3)$. We note that the interference function used in RTA [9], $W_i(t, d)$, is based on two different parameters, t for selecting the instances of tasks activated before or at t and d for the ones with a absolute deadline occurring before or at d (where $t < d$).

5. Conclusion and perspectives

In this paper, we have presented a new feasibility analysis, for tasks with offset (transactions) scheduled under EDF priorities. The analysis technique is based on the processor demand criteria and allows testing the global feasibility of a system with a pseudo-polynomial complexity. We have also reduced the complexity (execution time) of the algorithm, by proposing an efficient implementation technique, that consist in pre-calculating and storing statically the processor demand bound function and then use it to deduce the demand caused by a system for any time t. It is important to note that this pseudo-polynomial method, that we propose in this article, gives a necessary and sufficient schedulability test for a system of tasks with offset with EDF. In our knowledge, the other pseudo-polynomial tests, in the context of fixed or dynamic priorities, are RTA-based but provide only sufficient (non exact) schedulability test.

We see that our global analysis technique, based on the processor demand has a significantly lower complexity than the one of the response time analysis proposed by palencia [9]. The reason of the difference in complexity is that our test doesn't provide the response time of the tasks, but only checks if all the deadlines are met. We don't know yet if the results provided by the two methods are similar i.e if a system is decided feasible by one method, is it also by the other method? The subject of our future work is to answer to this question.

The authors are grateful to the reviewers for their help in greatly improving the paper.

References

[1] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority preemptive scheduling: An historical perspective. *Real-Time Systems 8*, pages 129–154, 1995.

[2] S. Baruah. *The Uniprocessor Scheduling of Sporadic Real-Time Tasks*. Phd thesis, Department of Computer Science. The University of Texas at Austin., 1993.

[3] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *The International Journal of Time-Critical Computing Systems*, (17):5–22, 1999.

[4] S. Baruah, A. Mok, and L. Rosier. The preemptive scheduling of sporadic real-time tasks on one processor. *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, 1990.

[5] S. Baruah, L. Rosier, and R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor. *The Journal of Real-Time Systems 2*, 1990.

[6] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publisher, Boston, 1997.

[7] M. Dertouzos. Control robotics: the procedural control of physical processors. *Proceedings of the IFIP Congress*, pages 807–813, October 1974.

[8] J. P. Gutierrez and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. *Proc IEEE Real-time System Symposium (RTSS)*, (19), December 1998.

[9] J. P. Gutierrez and M. G. Harbour. Offset-based response time analysis of distributed systems scheduled under edf. *Euromicro conference on real-time systems. Porto, Portugal*, June 2003.

[10] J. Leung and M. Merril. A note on preemptive scheduling of periodic real-time tasks. *Information Processing Letters 3(11)*, 1980.

[11] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in real-time environnement. *Journal of ACM*, 1(20):46–61, October 1973.

[12] J. Maki-Turja and M. Nolin. Fast and tight response-times for tasks with offsets. *17th EUROMICRO Conference on Real-Time Systems IEEE Palma de Mallorca Spain*, July 2005.

[13] J. Maki-Turja and M. Nolin. Efficient implementation of tight response-times for tasks with offsets. *Real-Time Systems Journal, Springer Netherlands*, 16 February 2008.

[14] A. Mok and D.Chen. A multiframe model for real-time tasks. *Proceeding of the 17th Real-Time Systems Symposium, Washington*, pages 22–29, December 1996.

[15] R. Pellizzoni and G. Lipari. Feasibility analysis of real-time periodic tasks with offsets. *Real-Time Systems*, 30(1-2):105–128, 2005.

[16] A. Rahni, E. Grolleau, and M. Richard. New worst-case response time analysis technique for real-time transactions. *ISoLa Workshop On Leveraging Applications of Formal Methods, Verification and Validation Isola2007 Poitiers France*, December 12-14 2007.

[17] M. Spuri. Analysis of deadline scheduled real-time systems. *RR-2772, INRIA, France*, 1996.

[18] K. Tindell. Adding time-offsets to schedulability analysis. *Technical Report YCS 221, Dept of Computer Science, University of York, England*, January 1994.

[19] K. Traore, E. Grolleau, and F. Cottet. Shedulability analysis of serial transactions. *Real-Time and Network Systems RTNS'06 Poitiers France*, pages 141–149, May 30-31 2006.