New Approach for Deadline Calculation of Periodic, Sporadic and Aperiodic Real-time Software Tasks

Aicha Goubaa[1,2,3], Mohamed Khalgui[2,3], Georg Frey[1] and Zhiwu Li[4,5]
[1]*Automation and Energy Systems, Saarland University, 66123, Saarbrucken, Germany*
[2]*School of Electrical and Information Engineering, Jinan University, Zhuhai Campus, 519070, Zhuhai, China*
[3]*National Institute of Applied Sciences and Technology (INSAT), University of Carthage, 1080, Tunis, Tunisia*
[4]*Institute of Systems Engineering, Macau University of Science and Technology, 999078, Taipa, Macau, China*
[5]*School of Electro-Mechanical Engineering, Xidian University, 710071, Xi'an, China*

Keywords: Real-time System, Periodic Task, Sporadic Task, Aeriodic Task, Deadline Calculation.

Abstract: A real-time system must react to events from the controlled environment while executing specific tasks that can be periodic, aperiodic or sporadic. These tasks can be subjected to a variety of temporal constraints, the most important one is the deadline. Thus, a reaction occurring too late may be useless or even dangerous. In this context, the main problem of this study is how to configure feasible real-time system having both periodic, aperiodic and sporadic tasks. In this paper, we propose a new off-line approach that configures feasible scheduling of a combination of software real-time tasks while serving aperiodic tasks without jeopardizing schedulability of periodic and sporadic ones.

1 INTRODUCTION

In the last years, real-time systems have become the focus of much study as this new technology is very attached to our daily life and can be nearly found in all domains such as transport, consumable electronics, games, etc (Ben Meskina et al., 2018; Lakhdhar et al., 2019; Ghribi et al., 2018). A real-time system is a computer system where the correctness of a computation depends on both the logical correctness of the results, and the time at which the computation completes (Burns and Wellings, 2001; Lakhdhar et al., 2018; Mavrommati et al., 2016). Such system reacts to signals from its environment by executing specific tasks that can be periodic, aperiodic or sporadic. A periodic task is activated on a regular cycle and must adhere to its hard deadline. It is characterized by its arrival time, worst-case execution time (WCET), period, relative deadline, and maximum deadline. A sporadic task can arrive to the system at arbitrary points in time, but with defined minimum inter-arrival time between two consecutive invocations. It is characterized by its worst-case execution time, minimum inter-arrival time, relative deadline, and maximum deadline. These attributes are known before system execution. Additional information available on-line, is its arrival time and its absolute deadline. An aperiodic task is activated at random time to cope with external interruptions, and it is based upon soft deadline. Its arrival time is unknown at design time. It is characterized by its worst-case execution time, and relative deadline.

To provide design-time guarantees on timing constraints, different scheduling methodologies can be used, such as Rate Monotonic (RM) scheduling algorithm which was defined by Liu and Layland (Liu and Layland, 1973) where the priority of tasks is inversely proportional to their periods, and Earliest Deadline First scheduling algorithm (EDF) which at each instant in time chooses for execution the currently active job with the smallest deadline (Baruah and Goossens, 2004). EDF is an optimal scheduling algorithm on preemptive uniprocessors, in the following sense: if a collection of independent jobs (each one characterized by an arrival time, an execution requirement, and a deadline) can be scheduled (by any algorithm) such that all the jobs complete by their deadlines, then the EDF will schedule this collection of jobs such that all of them complete by their deadlines. On the other hand, if a set of tasks is not schedulable under EDF, then no other scheduling algorithm can feasibly schedule this task set.

Enforcing timeliness constraints is necessary to maintain correctness of a real-time system. In order to

Goubaa, A., Khalgui, M., Frey, G. and Li, Z.
New Approach for Deadline Calculation of Periodic, Sporadic and Aperiodic Real-time Software Tasks.
DOI: 10.5220/0009831104520460
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 452-460
ISBN: 978-989-758-443-5

ensure a required real-time performance, the designer should predict the behavior of a real-time system by ensuring that all tasks meet their hard deadlines. Furthermore, scheduling both periodic, sporadic and aperiodic tasks in real-time systems is much more difficult than scheduling a single type of tasks. Thus, the development of real-time systems is not a trivial task because a failure can be critical for the safety of human beings (Wang et al., 2016). In this context, the considered problem is how to calculate the effective deadlines (hard and soft) of the different mixed tasks to guarantee that all tasks will always meet their deadlines while improving response times for aperiodic tasks.

Several research works have been done in recent years, focusing on real-time systems, such as those reported in (von der Brüggen et al., 2016; Shanmugasundaram et al., 2016; Gammoudi et al., 2015; Gammoudi et al., 2016; Gasmi et al., 2016; Wang et al., 2015a). Some of them (Gammoudi et al., 2015; Gammoudi et al., 2016; Gasmi et al., 2016) work on real-time schedulability without considering the deadlines analysis. Some other (von der Brüggen et al., 2016; Shanmugasundaram et al., 2016) seek to schedule tasks to respect energy constraints and consider that deadlines are given beforehand. Furthermore, these researches does not consider mixed tasks set. Moreover, techniques to calculate tasks' deadlines are seldom presented. For this reason, the studies that address this problem are few. The work reported in (Balbastre et al., 2007) presents a method that minimizes deadlines of periodic tasks only. The research in (Cervin et al., 2004) calculates new deadlines for control tasks in order to guarantee close loop stability of real-time control systems. On the other hand, several related works, such as in (Wang et al., 2014; Wang et al., 2015b) have chosen to manage the tasks of a real-time system by modifying either their periods or worst-case execution times (WCET). This orientation affects the performance of the system, since increasing the periods degrades the quality of the offered services, and decreasing the WCET increases the energy consumption.

We propose in this paper a new approach that configures feasible scheduling of software tasks of various types (periodic, sporadic and aperiodic) and constraints (hard and soft) in the context of dynamic-priority, preemptive, uniprocessor scheduling. The calculation of deadlines is performed off-line on the hyper-period which is the lowest common multiple (LCM) of the periodic tasks' periods (Ripoll and Ballester-Ripoll, 2012). In this paper, we suppose that the maximum number of occurrences of aperiodic tasks in a given interval of time is a random variable with a Poisson distribution which is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time. First, a periodic server is created to serve aperiodic tasks. A periodic server is a service task invoked periodically to execute aperiodic ones with: (i) a period which is calculated in such a way that the periodic execution of the server is repeated as many times as the maximum number of aperiodic tasks occurrences in the hyper-period, and (ii) a capacity which is the allowed computing time in each period and it is calculated based on unused processing time by a given set of periodic and sporadic tasks in the hyper-period in a such way aperiodic task execution should not jeopardize schedulability of periodic and sporadic tasks. Then, this approach calculates aperiodic tasks soft deadlines while supposing that an aperiodic task, with the smallest WCET, gets the highest priority. Second, this approach computes hard deadlines for periodic and sporadic tasks. In fact, for each periodic or sporadic task, it calculates the deadlines of its jobs that occur on the hyper-period based on (i) the maximum cumulative execution time requested by periodic and sporadic tasks that have to be executed before the considered job on the hyper-period, and (ii) the maximum cumulative execution time requested by aperiodic tasks that may occur before this job. For each periodic or sporadic task, the maximum among its calculated jobs deadlines will be its relative deadline. Thus, at runtime, even if an aperiodic task occurs, the periodic and sporadic tasks will certainly respect their deadlines and the response time of aperiodic task is improved as the invocation of aperiodic task execution is considered when calculating hard deadlines.

We note that most of existing studies working on real-time schedulability, address separately periodic, sporadic or aperiodic tasks but not together. Thus, the originality of this work compared with related studies is that it

- Deals with real-time tasks of various types and constraints simultaneously,
- Parameterizes periodic server to execute aperiodic tasks,
- Calculates soft deadlines of aperiodic tasks,
- Calculates periodic and sporadic tasks hard deadlines which will be certainly respected online,
- Improves response times of aperiodic tasks which can lead to a significant improvement of the system performance,
- Presents new tool called GIGTHIS-TOOL to evaluate the proposed solution.

The remainder of the paper is organized as follows. Section 2 presents the computational model and the considered assumptions. Section 3 explains in details the proposed approach to obtain valid deadlines. Section 4 presents the developed tool, and illustrates the approach on a case study while evaluating its efficiency. Finally, Section 5 summarizes this paper and provides directions for a future work.

2 PROBLEM FORMALIZATION

In this section, we present a formal description of a real-time system. We present in addition the different tasks models.

2.1 System Model

It is assumed in this work that a real-time system Π is defined as having three task sets: (i) the first, denoted $\mathcal{P}$, containing n periodic software tasks, i.e., $\mathcal{P} = \{\tau_1^0, ..., \tau_n^0\}$. We suppose that all these tasks are activated at $t = 0$. (ii) The second, denoted $\mathcal{S}$, containing m sporadic software tasks, i.e., $\mathcal{S} = \{\tau_1^1, ..., \tau_m^1\}$, and (iii) the third, denoted $\mathcal{A}$, containing k aperiodic software tasks, i.e., $\mathcal{A} = \{\tau_1^2, ..., \tau_k^2\}$.

2.2 Periodic Task Model

Each periodic task τ_i^0, $i \in [1,...,n]$, in $\mathcal{P}$ is characterized by: (i) a release time R_i^0 which is the time at which a task becomes ready for execution (Buttazzo, 2011), (ii) a worst-case execution time (WCET) C_i^0, (iii) a period P_i^0, (iv) a relative deadline D_i^0 to be calculated, and (v) a maximum relative deadline $Dmax_i^0$.

Each periodic task τ_i^0 produces an infinite sequence of identical activities called jobs τ_{ij}^0 (Buttazzo, 2011), where j is a positive integer. Each job τ_{ij}^0 is described by: (i) a release time r_{ij}^0, (ii) a relative deadline d_{ij}^0, and (iii) an end execution time E_{ij}^0. We note that

$$D_i^0 = max\{d_{ij}^0\} \quad (1)$$

where $i \in [1,...,n]$.

Finally, we denote by HP the hyper-period which is the lowest common multiple (LCM) of the periodic tasks' periods.

$$HP = LCM\{P_i^0\} \quad (2)$$

where $i \in [1,...,n]$.

2.3 Sporadic Task Model

Each sporadic task τ_e^1, $e \in [1,...,m]$, is defined by: (i) a release time R_e^1, (ii) a worst-case execution time C_e^1, (iii) a relative deadline D_e^1, (iv) a period P_e^1 which measures the minimum interval between the arrival of two successive instances of a task τ_e^1, and (v) a maximum relative deadline $Dmax_e^1$ (defined by users).

Each sporadic task τ_e^1 produces an infinite sequence of jobs τ_{ef}^1, where f is a positive integer. Each job τ_{ef}^1 is described by: (i) a release time r_{ef}^1, (ii) a relative deadline d_{ef}^1, and (iii) end execution time E_{ef}^1.

$$D_e^1 = max\{d_{ef}^1\} \quad (3)$$

where $e \in [1,...,m]$.

2.4 Aperiodic Task Model

Each aperiodic task τ_o^2, $o \in [1,...,k]$, is defined by: (i) a worst-case execution time C_o^2, and (ii) a relative soft deadline D_o^2. An aperiodic task can arrive in a completely random way. Thus, we model this number by the Poisson distribution with a parameter λ. We note by OC the maximum number of aperiodic tasks' occurrences estimated on the hyper-period.

Let NPS be a periodic server that behaves much like a periodic task, but created to execute aperiodic tasks. It is defined by: (i) a period P^s, and (ii) a capacity C^s. These parameters will be calculated to meet time requirements of aperiodic tasks.

2.5 Problem: Feasible Scheduling of Real-time Tasks with Various Types

The problem to be treated in this paper is how to parameterize the feasible scheduling of real-time tasks with various types and constraints in the context of dynamic-priority, preemptive, uniprocessor scheduling. Our scheduling problem contains two subproblems:

- Configuring aperiodic tasks: the execution of the aperiodic task must not jeopardize the schedulability of periodic and sporadic tasks, then the capacity of the NPS server must not be greater than the unused processing time by periodic and sporadic tasks. In addition, a soft deadline is calculated for each aperiodic task to allow that an aperiodic task with the smallest WCET will be served before another with high WCET.
- Real-time periodic and sporadic tasks' scheduling: during each hyper-period, each periodic or sporadic job has to be completed before the absolute deadline using the EDF scheduling algorithm

even if an aperiodic task is executed. In fact, the cumulative execution time requested by aperiodic tasks must be token into consideration when calculating the tasks' deadlines. Thus, s an aperiodic task will be executed as soon as possible of its activation, and periodic and sporadic tasks will meet their deadlines. This constraint is given by

- For periodic jobs:

$$\forall i \in \{1,...,n\}, \text{ and } j \in \{1,...,\frac{HP}{P_i^0}], E_{ij}^0 \leq r_{ij}^0 + D_i^0 \quad (4)$$

- For sporadic jobs:

$$\forall e \in \{1,...,m\}, \text{ and } f \in \{1,...,\left\lceil \frac{HP}{P_e^1} \right\rceil], E_{ef}^1 \leq r_{ef}^1 + D_e^1 \quad (5)$$

In what follows, it is always considered that $i \in [1...n]$, $e \in [1...m]$, $o \in [1...k]$, $j \in [1...\frac{HP}{P_i^0}]$, where $\frac{HP}{P_i^0}$ denotes the number of jobs produced by task τ_i on hyper-period HP and $f \in [1...\lceil \frac{HP}{P_i^0} \rceil]$.

3 CONTRIBUTION: NEW SOLUTION FOR DEADLINES CALCULATION

3.1 Motivation

We deliver an off-line approach, presented in Figure 1, that computes hard deadlines for periodic and sporadic tasks and soft deadlines for aperiodic ones while improving their response time. This method consists of two phases:

- The first one defines the *NPS* server which serves periodically aperiodic tasks. In fact, the server can be accounted for in periodic task schedulability analysis, it has a period P^s and a capacity C^s. Then, it calculates aperiodic tasks soft deadlines while supposing that an aperiodic task, with the smallest WCET, gets the highest priority.
- The second one calculates hard deadlines of periodic and sporadic tasks ensuring real-time system feasibility while considering the invocation of aperiodic task execution, i.e., while considering the maximum cumulative execution time requested by aperiodic tasks that may occur before periodic and sporadic jobs on the hyper-period. Thus, at runtime, even if an aperiodic task occurs, the periodic and sporadic tasks will certainly respect their deadlines and the response time of aperiodic task is improved.

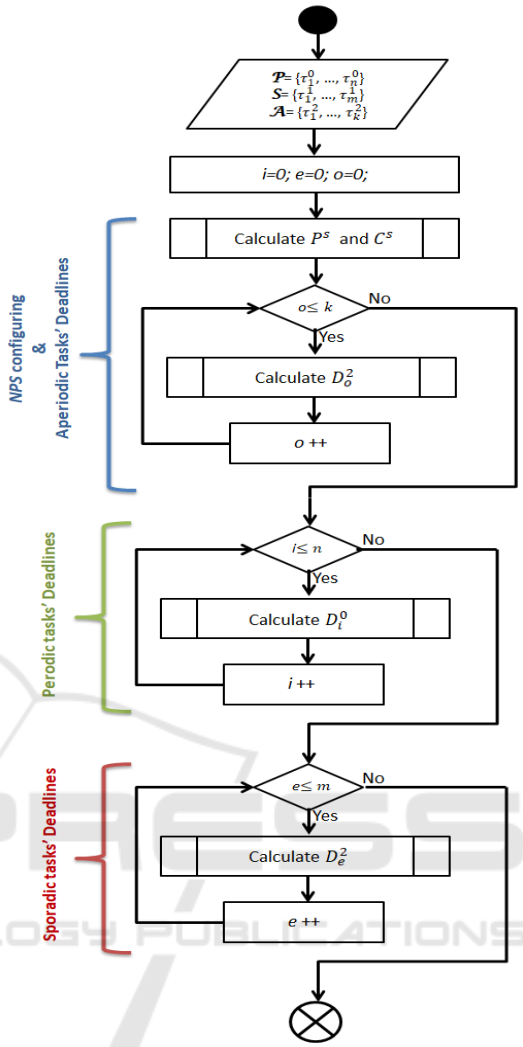

Figure 1: New methodology of deadlines calculation.

3.2 Case Study

We present in this section a case to be studied in the paper. We consider a real-time system to control temperature and humidity in a cold room. A temperature controller reads the temperature every 8s. If the measured value is not in the desired temperature interval, then a temperature regulator is activated to correct this problem. A humidity controller reads the humidity every 20s. If the measured value is not in the desired humidity interval, then a humidity regulator is activated to correct this problem. The measured values are displayed in a digital display every 5s. This system, noted Π, is powered by the battery that its status is checked at most once every 20s. This system is implemented with the following tasks presented in Table 1:

Thus, Π is implemented by three sets:

Table 1: System tasks.

Task	Fonction	WCET	Period	Maximum deadline
τ_1^0	displays temperature	1	5	6
τ_2^0	reads the temperature	2	8	10
τ_3^0	mesures humidity	3	20	18
τ_1^1	checks the battery status	2	20	23
τ_1^2	adjusts temperature	2		
τ_2^2	adjusts humidity	1		

$\mathcal{P} = \{\tau_1^0, \tau_2^0, \tau_3^0\}$, $\mathcal{S} = \{\tau_1^1\}$ and $\mathcal{A} = \{\tau_1^2, \tau_2^2\}$.

We have, $HP = LCM\{5, 8, 20\} = 40s$.

Let's suppose that the parameter λ of the Poisson distribution is equal to 0.5 occurences in 10 seconds. Thus, in the hyper-period we have $\frac{HP}{10} \times \lambda = \frac{40}{10} \times 0.5 = 2$ occurences, i.e., $OC = 2$.

3.3 Proposed Approach

In this section, we present the solution that we propose to extend. This solution is mainly based on the calculation of effective deadlines of mixed tasks set in order to ensure that the system will run correctly and to satisfy the real-time feasibility.

3.3.1 Parameterizing Aperiodic Tasks

As mentioned previously, aperiodic tasks will be run periodically by the periodic server NPS (P^s, C^s). As, OC is the maximum number of aperiodic tasks' occurrences estimated on the hyper-period, then, NPS must be activated OC times to serve all possible activations of aperiodic tasks that may occur. Thus, its period is calculated as bellow

$$P^s = \lfloor \frac{HP}{OC} \rfloor \tag{6}$$

Moreover, aperiodic tasks are scheduled by utilizing unused processing time by a given set of periodic and spordic tasks in the hyper-period. Thus, the capacity of server is calculated as follows: first, we calculate the unused time by subtracting the maximum cumulative execution time requested by periodic and sporadic jobs from HP, and second we divide the obtained result by OC, i.e., the possible activation number, to affirm that in each period the same amount of execution time will be executed, hence the server capacity value.

$$C^s = \lceil \frac{HP - Q}{OC} \rceil \tag{7}$$

where, Q is the maximum cumulative execution time requested by periodic and sporadic jobs on the hyper-period HP.

$$Q = (\sum_{\tau_i^0 \in \mathcal{P}} (C_i^0 \times \frac{HP}{P_i^0})) + (\sum_{\tau_e^1 \in \mathcal{S}} (C_e^1 \times \lceil \frac{HP}{P_e^1} \rceil)) \tag{8}$$

By assuming that the aperiodic task with the smallest C_o^2 gets the highest priority, we calculate the deadlines D_o^2 as following

$$D_o^2 = \sum_{x=1}^{x=k} C_x^2 \times \alpha_x \tag{9}$$

where,

$$\alpha_x = \begin{cases} 1 \text{ if } (C_o^2 > C_x^2) \text{ or } (C_o^2 = C_x^2 \text{ and } o \geq x), \\ 0 \text{ else.} \end{cases} \tag{10}$$

Running Example 1: To solve the problem of the control system of the case study, the first step is to configure the periodic server.

The periodic server parameters P^s and C^s are computed respectively as following:

According to Equation (6), $P^s = \lfloor \frac{40}{2} \rfloor = 20$

According to Equation (8), $Q = 1 \times 8 + 2 \times 5 + 3 \times 2 + 2 \times 2 = 28$

According to Equation (7), $C^s = \lfloor \frac{40 - 28}{2} \rfloor = 6$

After that, we calculate aperiodic tasks' deadlines. Let's take for example τ_1^2. According to Equation (9)

$$\begin{aligned} D_1^2 &= C_1^2 \times \alpha_1 + C_2^2 \times \alpha_2 \\ &= 2 \times 1 + 1 \times 1 = 3 \end{aligned}$$

3.3.2 Parameterizing Periodic and Sporadic Tasks:

At the peak of activity, a sporadic task τ_e runs at each P_e^1. In this case, we can estimate the value r_{ef}^1 of each job τ_{ef}^1. Therefore, to calculate the deadline of a sporadic task, we follow the same procedure of a periodic task deadline calculation. For that, we unify the notation of periodic and sporadic tasks by $\tau_{i_1}(R_{i_1}, C_{i_1}, P_{i_1}, D_{i_1}, Dmax_{i_1})$, where i_1 $in[1, ..., n+m]$, also for these parameters. For example, let's consider a system with two tasks: a periodic task $\tau_1^0(R_1^0, C_1^0, P_1^0, D_1^0, Dmax_1^0)$ and a sporadic task $\tau_1^1(R_1^1, C_1^1, P_1^1, D_1^1, Dmax_1^1)$,

then they becomes $\tau_1(R_1, C_1, P_1, D_1, Dmax_1)$ and $\tau_2(R_2, C_2, P_2, D_2, Dmax_2)$.

This solution allows the calculation of deadlines of a task τ_{i_1}. We denote by $\Delta_{i_1 j_1}$ the job quantity, coming from periodic and sporadic jobs, to be executed before the job $\tau_{i_1 j_1}$. In other words, $\Delta_{i_1 j_1}$ is the maximum cumulative execution time requested by jobs whose (i) maximum absolute deadlines are less than that of job $\tau_{i_1 j_1}$. This condition is denoted by *C1*, or (ii) maximum absolute deadlines are equal to that of job $\tau_{i_1 j_1}$ and arrival times are less than that of job $\tau_{i_1 j_1}$ or their indices are less than i, i.e., we apply the strategy of first in first out (FIFO) by assuming that the task with the smallest index is the one that comes at the beginning. This condition is denoted by *C2*. $\Delta_{i_1 j_1}$ is given by

$$\Delta_{i_1 j_1} = \sum_{\tau_l \in \mathcal{P} \cup \mathcal{S}} (C_l \times \beta_l^{i_1 j_1}) \tag{11}$$

where $\beta_l^{i_1 j_1}$ is the number of jobs produced by a periodic or sporadic task τ_l to be executed before $\tau_{i_1 j_1}$, represented as

$$\beta_l^{i_1 j_1} = \begin{cases} \left\lceil \frac{(j_1 - 1)P_{i_1} + Dmax_{i_1} - Dmax_l}{P_l} \right\rceil \text{ if } (C1 = true \\ \text{and } C2 = false), \\ \left\lceil \frac{(j_1 - 1)P_{i_1} + Dmax_{i_1} - Dmax_l}{P_l} \right\rceil + 1 \\ \text{if } (C1 = true \text{ and } C2 = true), \\ 1 \text{ if } (C1 = false \text{ and } C2 = true), \\ 0 \text{ else.} \end{cases} \tag{12}$$

The value $d_{i_1 j_1}$ that guarantees the feasibility of this job takes the form

$$d_{i_1 j_1} = \begin{cases} \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_{i_1} + \Delta_{i_1 j_1} - r_{i_1 j_1} \\ \text{if } \Delta_{i_1 j_1} > r_{i_1 j_1}, \\ \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_{i_1} \text{ else.} \end{cases} \tag{13}$$

The deadline D_{i_1} of task τ_{i_1} is expressed by

$$D_{i_1} = max\{d_{i_1 j_1}\} \tag{14}$$

Finally, D_{i_1} is the fixed deadline for τ_{i_1}.

Running Example 2: We move to periodic and sporadic tasks' deadlines calculation. As mentioned previously, we unify the notation of periodic and sporadic tasks as following: τ_1^0 becomes τ_1, τ_2^0 becomes τ_2, τ_3^0 becomes τ_3 and τ_1^1 becomes τ_4.

As an example, we take the calculation of deadline D_3^0 for task τ_3^0, i.e., D_3 for the task τ_3. The number of jobs of task τ_3 in the hyper-period HP is $\frac{HP}{P_3} = \frac{40}{20} = 2$ jobs.

Job τ_{31}:

First of all, we calculate the job quantity Δ_{31}. According to Equation (11), we have to calculate β_1^{31}, β_2^{31}, β_3^{31} and β_4^{31} as indicated in Equation((12)).

$$\beta_1^{31} = \left\lceil \frac{(1-1)P_3 + Dmax_3 - Dmax_1}{P_1} \right\rceil = \left\lceil \frac{(1-1)20 + 18 - 6}{5} \right\rceil = 3$$

$$\beta_2^{31} = \left\lceil \frac{(1-1)P_3 + Dmax_3 - Dmax_2}{P_2} \right\rceil = \left\lceil \frac{(1-1)20 + 18 - 10}{8} \right\rceil = 1$$

$$\beta_3^{31} = \left\lceil \frac{(1-1)P_3 + Dmax_3 - Dmax_3}{P_3} \right\rceil = \left\lceil \frac{(1-1)20 + 18 - 18}{20} \right\rceil = 0$$

For the task τ_4, the first job τ_{41} has: (i) an absolute deadline greater than that of τ_{31}, (ii) an arrival time r_{41} equal to that of τ_{31}, and (iii) an index greater than that of τ_{31}. According to Equation (12), $C1 = false$ and $C2 = false$. Therefore, $\beta_4^{31} = 0$.

According to Equation (11), Δ_{31} is calculated as following

$$\Delta_{31} = \sum_{\tau_l \in \mathcal{P} \cup \mathcal{S}} C_1 \times \beta_1^{31} + C_1 \times \beta_2^{31} + C_3 \times \beta_3^{31} + C_4 \times \beta_4^{31}$$
$$= 1 \times 3 + 2 \times 1 + 2 \times 0 + 3 \times 0 = 5$$

We have $r_{31} = 0$, so $\Delta_{31} > r_{31}$ and we have

$$\sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) = C_1^2 \times \lceil \frac{Pi_1}{P^s} \rceil + C_2^2 \times \lceil \frac{Pi_1}{P^s} \rceil$$
$$= 1 \times \lceil \frac{20}{20} \rceil + 2 \times \lceil \frac{20}{20} \rceil = 3$$

Thus, according to Equation (13),

$$d_{31} = \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_3 + \Delta_{31} - r_{31}$$
$$= 3 + 3 + 5 = 11$$

Job τ_{32}:

First of all, we calculate the job quantity Δ_{32}. According to Equation (11), we have to calculate β_1^{32}, β_2^{32}, β_3^{32} and β_4^{32} as indicated in Equation((12)).

$$\beta_1^{32} = \left\lceil \frac{(2-1)P_3 + Dmax_3 - Dmax_1}{P_1} \right\rceil = \left\lceil \frac{20 + 18 - 6}{5} \right\rceil = 7$$

$$\beta_2^{32} = \left\lceil \frac{(2-1)P_3 + Dmax_3 - Dmax_2}{P_2} \right\rceil = \left\lceil \frac{20+18-10}{8} \right\rceil = 4$$

$$\beta_3^{32} = \left\lceil \frac{(2-1)P_3 + Dmax_3 - Dmax_3}{P_3} \right\rceil = \left\lceil \frac{20+18-18}{20} \right\rceil = 1$$

$$\beta_4^{32} = \left\lceil \frac{(2-1)P_3 + Dmax_3 - Dmax_4}{P_4} \right\rceil = \left\lceil \frac{20+18-23}{20} \right\rceil = 1$$

According to Equation (11), Δ_{32} is calculated as following

$$\Delta_{32} = \sum_{\tau_l \in \mathcal{P} \cup \mathcal{S}} C_1 \times \beta_1^{32} + C_1 \times \beta_2^{32} + C_3 \times \beta_3^{32} + C_4 \times \beta_4^{32} = 1 \times 7 + 2 \times 4 + 3 \times 1 + 2 \times 1 = 20$$

We have $r_{32} = 20$, then $\Delta_{32} = r_{32}$ and we have

$$\sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) = 3$$

Thus, according to Equation (13),

$$d_{31} = \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_3 = 3 + 3 = 6$$

Finally, we calculate the deadline D_3 of the task τ_3 as bellow

$$D_3 = max\{d_{31}, d_{32}\} = max\{11, 6\} = 11$$

4 IMPLEMENTATION

4.1 Developed Environment: GIGTHIS-TOOL

GIGTHIS-TOOL is a new simulator that applies the services of the proposed methodology. This tool is an open source environment that: either add a new system; i.e., sets of periodic, sporadic and aperiodic tasks or generate the different tasks sets randomly, and then apply the proposed approach while:(i) calculating the hyper-period, (ii) generating the maximum occurrence number of the aperiodic tasks, (iii) calculating the capacity and the period of the *NPS* server, and (iv) calculating the deadlines of all tasks by applying the formulas of the proposed methodology. GIGTHIS-TOOL can be simply used by designers to compute and display effective deadlines, with few clicks, in arranged tables and in short time. This project can be a future reference for industrial partners who will be focusing on various real-time applications design.

4.2 Case Study Results

After completing the execution of the proposed approach, the calculated effective deadlines of the different tasks are given in Table 2.

Table 2: Tasks' calculated deadlines.

Task	τ_1^0	τ_2^0	τ_3^0	τ_1^1	τ_1^2	τ_2^2
Calculated Deadline	4	6	11	16	3	1

The results of the case study by using GIGTHIS-TOOL[1] are stored in this link [2].

In order to test the validity of the obtained results, we use the Cheddar environment [3]. Figure 2 shows the scheduling of tasks after the execution of the proposed approach. We note that the real-time constraints are respected by the proposed methodology, and the response time of each aperiodic task is equal to its execution time, i.e, they are executed with the best response time.

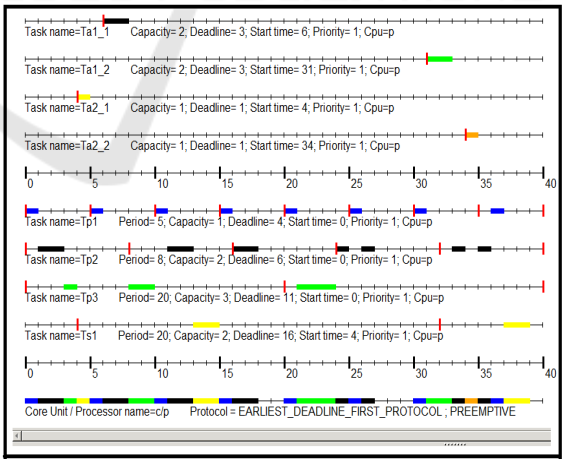

Figure 2: Scheduling of tasks after the execution of the proposed approach.

[1] As writing index and exponent is not allowed in GIGTHIS-TOOL (as well as cheddar tool), then the tasks notation becomes as follows: Tp_i for a periodic task, Ts_e for a sporadic task and Ta_o for an aperiodic one.

[2] https://projects-lisi-lab.wixsite.com/gigthistool

[3] http://beru.univ-brest.fr/ singhoff/cheddar/

4.3 Performance Evaluation

We have randomly generated instances with 10 to 50 periodic and sporadic tasks. We compare the proposed approach with the work reported in (Balbastre et al., 2007), where the critical scaling factor (CSF) algorithm is developed. We focus on the reduction rate of the calculated deadlines compared to maximum deadlines.

Figure 3 shows that the reduction rates of deadlines by using (Balbastre et al., 2007) are smaller than those by using the proposed work. We conclude that the rate of reduction of deadlines in (Balbastre et al., 2007) can be improved. Hence, the gain is offered by the proposed approach. Moreover, the gain is more significant when increasing the number of tasks. If 10 tasks are considered, then the gain is equal to (0.31-0.2) = 0.11, and if 50 tasks are considered, then the gain is equal to (0.61-0.35) = 0.26.

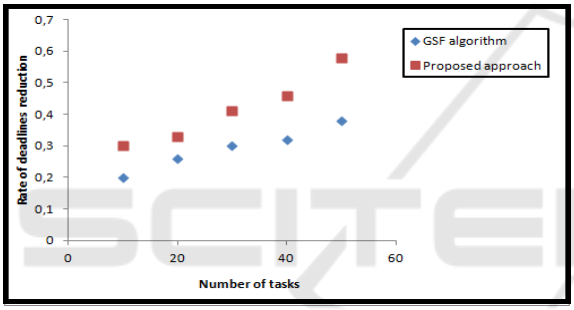

Figure 3: Rates of deadlines reduction in the case of the proposed approach and in the case of GSF algorithm.

The histogram in Figure 4 reflects the difference between the maximum deadlines and the calculated effective deadlines. In fact, the calculated deadlines of the proposed approach are decreased compared to the maximum ones of the 50 tasks. Thus, this reduction enhances the performance and the stability of the real-time system by finishing the execution of tasks at the perfect time.

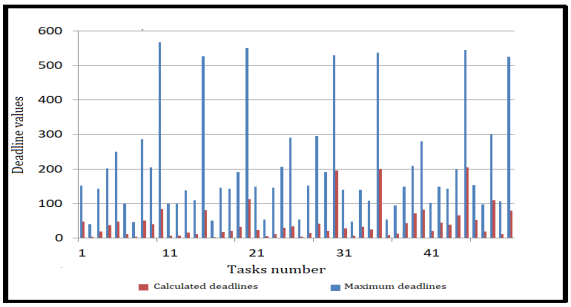

Figure 4: The difference between the maximum deadlines and the calculated deadlines.

As can be seen from the first graph presented in Figure 5, the *NPS* algorithm can provide a substantial reduction in average aperiodic response time compared to background service (BK), deferrable server (DS) and total bandwidth server (TBS). This algorithm provides the greatest improvement for short, frequent aperiodic tasks.

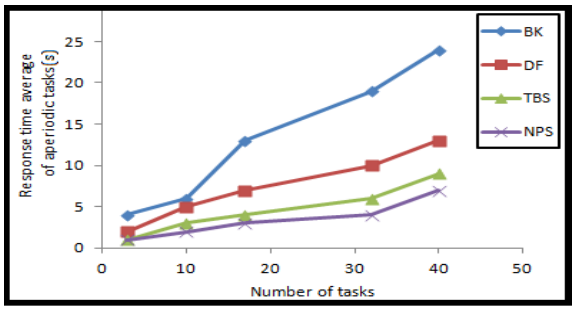

Figure 5: The improvement of aperiodic tasks response times.

We note that the proposed approach allows to reduce the response time, to reduce the calculation time for the reason that there is no need to waste time at doing schedulability tests, to guarantee the meeting of aperiodic tasks deadlines without jeopardizing schedulability of periodic and sporadic tasks and thus improves the overall performance of the real-time system.

5 CONCLUSION

The contribution presented in this paper consists in a methodology that supports the deadline calculation of a mixed real-time system. By defining the specification such as temporal constraints the approach starts at the first step by building a periodic server to serve aperiodic tasks by identifying the server's period and capacity. Then, it computes the aperiodic tasks' deadlines by assuming that the one with the smallest WCET, gets the highest priority. Finally, this approach calculates the periodic and sporadic tasks deadlines while considering the invocation of aperiodic task execution. We have evaluated the performance of the proposed approach. The numerical results show that this methodology reduces the development time by computing the deadlines for periodic and sporadic tasks to be certainly respected without any feasibility analysis of the device, and allows to minimize the response time for aperiodic tasks. As a future work, we aim to extend our approach by considering distributed architecture and other constraints e.g. energy and memory constraints.

REFERENCES

Balbastre, P., Ripoll, I., and Crespo, A. (2007). Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. *IEEE Transactions on computers*, 57(1):96–109.

Baruah, S. and Goossens, J. (2004). Scheduling real-time tasks: Algorithms and complexity. *Handbook of scheduling: Algorithms, models, and performance analysis*, 3.

Ben Meskina, S., Doggaz, N., Khalgui, M., and Li, Z. (2018). Reconfiguration-based methodology for improving recovery performance of faults in smart grids. *J. Information Sciences*, 454-455:73–95.

Burns, A. and Wellings, A. J. (2001). *Real-TIme Systems and Programming Languages. 3rd*. Pearson Education.

Buttazzo, G. C. (2011). *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. Springer Science & Business Media.

Cervin, A., Lincoln, B., Eker, J., Arzén, K.-E., and Buttazzo, G. (2004). The jitter margin and its application in the design of real-time control systems. In *Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems and Applications*, pages 1–9. Gothenburg, Sweden.

Gammoudi, A., Benzina, A., Khalgui, M., and Chillet, D. (2015). New pack oriented solutions for energy-aware feasible adaptive real-time systems. In *International Conference on Intelligent Software Methodologies, Tools, and Techniques*, pages 73–86. Springer.

Gammoudi, A., Benzina, A., Khalgui, M., Chillet, D., and Goubaa, A. (2016). Reconf-pack: A simulator for reconfigurable battery-powered real-time systems.

Gasmi, M., Mosbahi, O., Khalgui, M., Gomes, L., and Li, Z. (2016). R-node: New pipelined approach for an effective reconfigurable wireless sensor node. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(6):892–905.

Ghribi, I., Ben Abdallah, R., Khalgui, M., Li, Z. andAlnowibet, K., and Platzne, M. (2018). R-codesign: Codesign methodology for real-time reconfigurable embedded systems under energy constraints. *IEEE Access*, 6:14078–14092.

Lakhdhar, W., Mzid, R., Khalgui, M., Li, Z., Frey, G., and Al-Ahmari, A. (2018). Multiobjective optimization approach for a portable development of reconfigurable real-time systems: From specification to implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(3):623–637.

Lakhdhar, W., Mzid, R., Khalgui, M., Li, Z., Frey, G., and Al-Ahmari, A. (2019). Multi-objective optimization approach for a portable development of reconfigurable real-time systems: From specification to implementation. *IEEE Trans. Syst., Man, Cybern., Syst.*, 49:623–637.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61.

Mavrommati, A., Schultz, J., and Murphey, T. D. (2016). Real-time dynamic-mode scheduling using single-integration hybrid optimization. *IEEE Transactions on Automation Science and Engineering*, 13(3):1385–1398.

Ripoll, I. and Ballester-Ripoll, R. (2012). Period selection for minimal hyperperiod in periodic task systems. *IEEE Transactions on Computers*, 62(9):1813–1822.

Shanmugasundaram, M., Kumar, R., and Kittur, H. M. (2016). Performance analysis of preemptive based uniprocessor scheduling. *International Journal of Electrical and Computer Engineering*, 6(4):1489.

von der Brüggen, G., Huang, W.-H., Chen, J.-J., and Liu, C. (2016). Uniprocessor scheduling strategies for self-suspending task systems. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 119–128. ACM.

Wang, X., Khemaissia, I., Khalgui, M., Li, Z., Mosbahi, O., and Zhou, M. (2014). Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks. *IEEE Transactions on Automation Science and Engineering*, 12(1):258–271.

Wang, X., Li, Z., and Wonham, W. (2015a). Dynamic multiple-period reconfiguration of real-time scheduling based on timed des supervisory control. *IEEE Transactions on Industrial Informatics*, 12(1):101–111.

Wang, X., Li, Z., and Wonham, W. (2015b). Dynamic multiple-period reconfiguration of real-time scheduling based on timed des supervisory control. *IEEE Transactions on Industrial Informatics*, 12(1):101–111.

Wang, X., Li, Z., and Wonham, W. M. (2016). Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on des supervisory control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1082–1098.