Efficient Scheduling of Periodic, Aperiodic, and Sporadic Real-Time Tasks with Deadline Constraints

Aicha Goubaa[1,2,3(✉)], Mohamed Kahlgui[2,3], Frey Georg[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), Zhuhai 519070, 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, Taipa, Macau 999078, China
[5] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China

Abstract. A real-time system is an operating system that guarantees a certain functionality within a specified time constraint. Such system is composed of tasks of various types: periodic, sporadic and aperiodic. 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. This paper shows an approach for computing deadlines in uniprocessor real-time systems to guarantee real-time feasibility for hard-deadline periodic and sporadic tasks and provide good responsiveness for soft-deadline aperiodic tasks. An application to a case study and performance evaluation show the effectiveness of the proposed approach.

Keywords: Real-time feasibility · Periodic task · Sporadic task · Aeriodic task · Hard deadline · Soft deadline

Nomenclature

Π	Real-time system;
$\mathcal{P}$	Set of periodic tasks in Π;
$\mathcal{S}$	Set of sporadic tasks in Π;
$\mathcal{A}$	Set of aperiodic tasks in Π;
n	Number of periodic tasks in Π;
m	Number of sporadic tasks in Π;
k	Number of aperiodic tasks in Π;

M. van Sinderen et al. (Eds.): ICSOFT 2020, CCIS 1447, pp. 25–43, 2021.
https://doi.org/10.1007/978-3-030-83007-6_2

τ_i^0	Periodic task;
τ_{ij}^0	The jth job of τ_i^0;
τ_e^1	Sporadic task;
τ_{ef}^1	The fth job of τ_e^0;
τ_o^2	Aperiodic task;
R_i^0	Release time of τ_i^0;
r_{ij}^0	Release time of the jth job of τ_i^0;
C_i^0	Worst-case execution time of τ_i^0;
P_i^0	Period of τ_i^0;
D_i^0	Hard relative deadline of τ_i^0 to be determined;
d_{ij}^0	Relative deadline of τ_{ij}^0 to be determined;
ϕ_i^0	Degree of criticality of τ_i^0;
E_{ij}^0	End execution time of τ_{ij}^0;
R_e^1	Release time of τ_e^1;
r_{ef}^1	Release time of the fth job of τ_e^1;
C_e^1	Worst-case execution time of τ_e^1;
P_e^1	Minimum interval between the arrival of two successive instances of τ_e^1;
D_e^1	Hard relative deadline of τ_e^1 to be determined;
d_{ef}^1	Relative deadline of τ_{ef}^1 to be determined;
ϕ_e^1	Degree of criticality of τ_e^1;
E_{ef}^1	End execution time of τ_{ef}^1;
C_o^2	WCET of τ_o^2;
D_o^2	Soft deadline of τ_o^2 to be determined;
ϕ_o^2	Degree of criticality of τ_o^2;
C^s	Capacity of the NPS server;
P^s	Period of the NPS server;
HP	Hyper-period;
OC	Maximum number of aperiodic tasks' occurrences estimated on HP;
Q	Maximum cumulative execution time requested by periodic and sporadic jobs on HP;
τ_{i_1}	Periodic or sporadic task;
$\tau_{i_1 j_1}$	The j_1th job of $\tau_{i_1 j_1}$;
$\Delta_{i_1 j_1}$	Maximum cumulative execution time requested by periodic and sporadic jobs that have to be executed before $\tau_{i_1 j_1}$;
$\beta_l^{i_1 j_1}$	Number of jobs produced by a periodic or sporadic task τ_l to be executed before $\tau_{i_1 j_1}$.

1 Introduction

Nowadays, computer systems, to control real-time functions, are considered among the most challenging systems. As a consequence, real-time systems have become the focus of much study [4–6]. A real-time system is any system which has to respond to externally generated input stimuli within a finite and specified delay. The development of real-time systems is not a trivial task because a

failure can be critical for the safety of human beings [1–3]. Such system must react to events from the controlled environment while 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 a degree of criticality that defines its applicative importance. The degree of criticality is defined as the functional and operational importance of a task. 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 a degree of criticality that defines its applicative importance. 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, relative deadline, and a degree of criticality that defines its applicative importance.

Real-time scheduling has been extensively studied in the last three decades. These studies propose several Feasibility Conditions for the dimensioning of real-time systems. These conditions are defined to enable a designer to grant that timeliness constraints associated with an application are always met for all possible configurations. In this paper, Two main classical scheduling are generally used in real-time embedded systems: RM and EDF. EDF is a dynamic scheduling algorithm used in real-time operating systems [8]. 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. Rate Monotonic (RM) for fixed priorities and Earliest, it was defined by Liu and Layland [7] where the priority of tasks is inversely proportional to their periods.

Enforcing timeliness constraints is necessary to maintain correctness of a real-time system. In order to 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 [19]. 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.

The major contribution of this work is a methodology defined in the context of dynamic priority preemptive uniprocessor scheduling to achieve real-time fea-

sibility of a software system. Differently from earlier work [22], which is based on maximum deadlines, the deadline calculation in the current work is based on the degree of criticality of tasks and on their periods. In fact, as the degree of criticality is defined as the functional and operational importance of a task, we consider that an important task must be executed ahead, i.e., that its relative deadline must be well defined to reinforce its execution while using the EDF scheduling algorithm. The calculation of deadlines is done off-line on the hyper-period which is the lowest common multiple (LCM) of the periodic tasks' periods [20]. 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. This proposed approach 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 (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 highest degree of criticality, gets the highest priority to be executed. 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. 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.

The remainder of the paper is organized as follows. Section 2 discusses the related studies. Section 3 presents a computational model, assumptions, and problem formulation. Section 4 gives the proposed scheduling method. Section 5 presents a case study for evaluating our method. Finally, Sect. 6 summarizes this paper with our future work.

2 Related Studies

In this section, we present the related works that deal with real-time systems and scheduling policies.

Several works deal with the synthesis problem of real-time systems. The correctness of such systems depends both on the logical result of the computation and the time when the results are produced. Thus enforcing timeliness constraints is necessary to maintain correctness of a real-time system. In this context, many approaches have been carried out in the area of schedulability analysis for meeting real-time requirement [9–14,26]. Some of them [11–13] work on real-time schedulability without considering the deadlines analysis. Some other [9,10] seek to schedule tasks to respect energy constraints and consider that deadlines are given beforehand. Pillai and Shin [26] propose an optimal algorithm for computing the minimal speed that can make a task set schedulable. 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 [15] presents a method that minimizes deadlines of periodic tasks only. The research in [18] 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 [16,17] 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.

Furthermore, the research works reported in [23–25] take into account the energy requirements without considering the deadlines analysis, as long as they are given beforehand. Indeed, in these researches, the authors seek to schedule tasks to respect energy constraints. In addition, it is done online, which can be heavy and expensive.

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.

3 Assumptions and System Formalization

3.1 System Model

It is assumed in this work that a real-time system Π deals with a combination of mixed sets of tasks and constraints: periodic and sporadic tasks with hard constraints, and soft aperiodic tasks. Thus, Π is defined as having three task sets as presented in Fig. 1.

We assume that all periodic tasks are simultaneously activated at time $t = 0$;

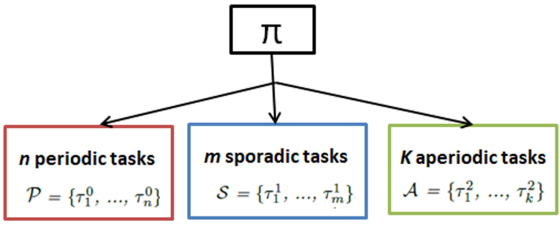

Fig. 1. Π's tasks sets.

3.2 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 [27], (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 degree of criticality phi_i^0. Figure 2 depicts the task parameters:

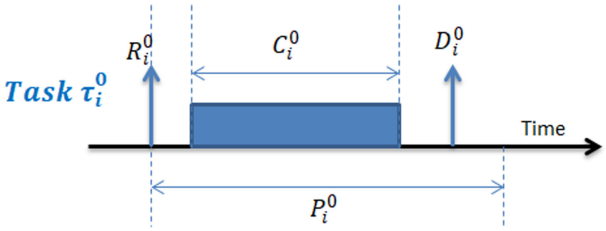

Fig. 2. Periodic task parameters.

Each periodic task τ_i^0 produces an infinite sequence of identical activities called jobs τ_{ij}^0 [27], 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

$$D_i^0 = max\{d_{ij}^0\} \tag{1}$$

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.

$$HP = LCM\{P_i^0\} \tag{2}$$

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

3.3 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 degree of criticality phi_e^1

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. Figure 3 depicts the sporadic task's jobs parameters:

$$D_e^1 = max\{d_{ef}^1\} \tag{3}$$

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

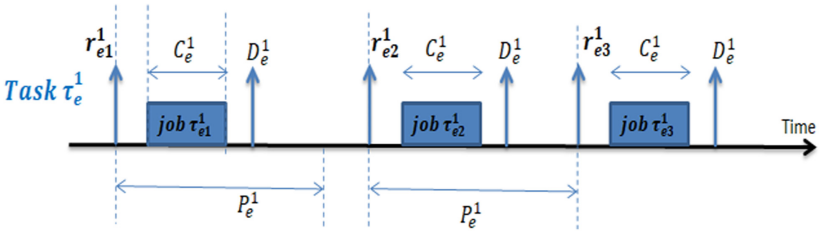

Fig. 3. Sporadic task parameters.

3.4 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, (ii) a relative soft deadline D_o^2, and (iii) a degree of criticality phi_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.

3.5 Problem: Feasible Scheduling of Real-Time Tasks with Various Types

We undertake a real-time system which is composed of mixed tasks sets with various constraints. Thus, the considered problem is how to configure 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. To ensure that this system runs correctly, it is necessary to check whether it respects the following constraints:

– the execution of aperiodic tasks must occur during the unused processing time by a given set of periodic and sporadic tasks in the hyper-period in such way aperiodic task execution should not jeopardize schedulability of periodic and sporadic tasks. This constraint is given by

$$C^s \leq HP - Q$$

where, C^s is the capacity of the NPS server and Q is the maximum cumulative execution time requested by periodic and sporadic jobs on the hyper-period HP.

– 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:

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

  • For sporadic jobs:

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

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 $f \in [1...\lceil \frac{HP}{P_i^0} \rceil]$. In addition, we suppose that a ztask lower its value, higher the criticality.

4 Contribution: New Solution for Deadlines Calculation

4.1 Motivation

The proposed methodology deals with real-time tasks of various types and constraints simultaneously. This approach is divided into two phases as presented in Fig. 4:

- **First Phase:** consists on parameterizing the NPS server which is a service task, with a period P^s and a capacity C^s, invoked periodically to execute aperiodic tasks. Then, this approach calculates soft deadlines of aperiodic tasks while supposing that an aperiodic task, with highest degree of criticality, gets the highest priority. The NPS can provide a substantial reduction in the average response time of the aperiodic tasks.
- **Second Phase:** starts by calculating jobs'deadlines. In fact, for each periodic/sporadic task, it calculates the deadlines of its jobs that occur on the hyper-period based on the maximum cumulative execution time requested by i) other periodic/sporadic jobs that will occur before the considered periodic/sporadic job on the hyperperiod based on the degree of criticality, and ii) aperiodic tasks that may occur before periodic/sporadic job on the hyperperiod. Then, for each periodic/sporadic task, its deadline will be equal to the maximum of its jobs' deadlines. Thus, at runtime, even if an aperiodic task occurs, this methodology ensures certainly real-time system feasibility of periodic and sporadic tasks.

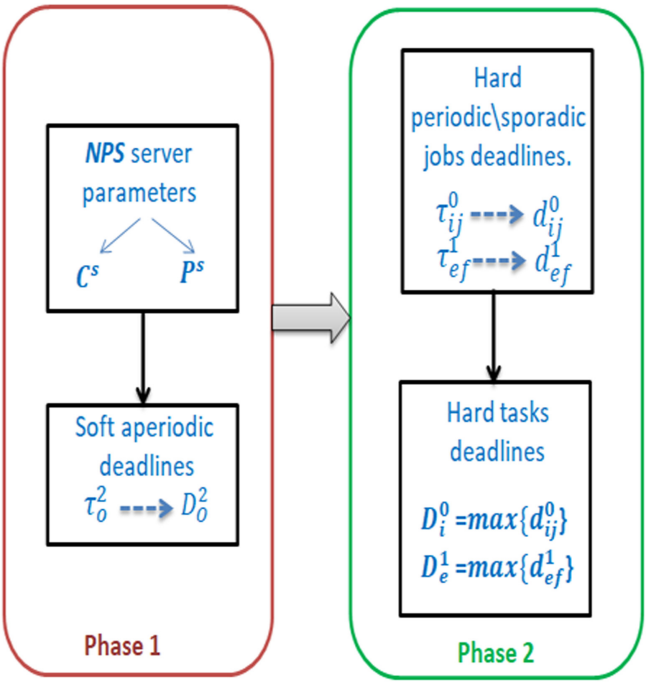

Fig. 4. New methodology of deadlines calculation.

4.2 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.

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 below

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

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 highest degree of criticality,i.e., the smallest value of ϕ_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,

$$\alpha_x = \begin{cases} 1 \text{ if } (\phi_o^2 \geq \phi_x^2), \\ 0 \text{ else.} \end{cases} \tag{10}$$

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 $\tau_{i_1}(R_{i_1}, C_{i_1}, P_{i_1}, D_{i_1}, \phi_{i_1})$, where i_1 $in[1, ..., n+m]$, also for these parameters. For example, let's consider a system with two tasks: a periodic task $\tau_1^0(R_1^0, C_1^0, P_1^0, D_1^0, \phi_1^0)$ and a sporadic task $\tau_1^1(R_1^1, C_1^1, P_1^1, D_1^1, \phi_1^1)$, then they becomes $\tau_1(R_1, C_1, P_1, D_1, \phi_1)$ and $\tau_2(R_2, C_2, P_2, D_2, \phi_2)$.

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 that have to be executed before each job $\tau_{i_1 j_1}$.

$\Delta_{i_1 j_1}$ is given by

$$\Delta_{i_1 j_1} = (j_1 - 1) \times C_{i_1} + \sum_{\tau_l^{\in} \mathcal{P} \cup \mathcal{S} and l \neq i} (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor - \beta_l^{i_1 j_1}) \times C_l \tag{11}$$

where $(j_1 - 1) \times C_{i_1}$ represents the cumulative execution time requested by the previous instances of τ_{i_1}, i.e., if we are working on the j_1th instance, then we are sure that there are $(j_1 - 1)$ instances that have already been executed, and $\sum_{\tau_l^{\in} \in \mathcal{P} \cup \mathcal{S} and l \neq i} (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor - \beta_l^{i_1 j_1}) \times C_l$ represents the cumulative execution time requested by the other tasks' jobs, where $\beta_l^{i_1 j_1}$ is an integer given by

$$\beta_l^{i_1 j_1} = \begin{cases} 0 \text{ if } (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor \times P_l < j_1 \times P_{i_1}) \text{ or } (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor \times P_l = j_1 \times P_{i_1} \text{ and } \phi_{i_1} > \phi_l) \\ 1 \text{ if } (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor \times P_l > j_1 \times P_{i_1}) \text{ or } (\lfloor \frac{j_1 \times P_{i_1}}{P_l} \rfloor \times P_l = j_1 \times P_{i_1} \text{ and } \phi_{i_1} < \phi_l) \end{cases} \tag{12}$$

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

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

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}.

New Solution for Deadline Calculation of Periodic, Sporadic and Aperiodic Real-Time Tasks: We can implement our approach by the algorithm below with complexity O(n).

We use the following functions: $NPS\_Parameters(HP, OC)$ which is a function that returns the NPS server parameters, and $H\_Dead\_Calc(\mathcal{P}, \mathcal{S})$ which a function that returns the periodic and sporadic hard deadlines. This function starts by computing jobs deadlines and then for each periodic/sporadic task, it calculates its fixed deadline to be equal to the maximum of its jobs' deadlines.

Algorithm 1. New method for deadline calculation.

Require: $\mathcal{P}$, $\mathcal{S}$,$\mathcal{A}$, OC
Ensure: D_i^0, D_e^1, D_o^2
function $NPS\_Parameters(HP, OC)$
  $P^s = \lfloor \frac{HP}{OC} \rfloor$
  $Q = (\sum_{\tau_i^0 \in \mathcal{P}} (C_i^0 \times \frac{HP}{P_i^0})) + (\sum_{\tau_e^1 \in \mathcal{S}} (C_e^1 \times \lceil \frac{HP}{P_e^1} \rceil))$
  $C^s = \lceil \frac{HP - Q}{OC} \rceil$
end function
for all $\tau_o^2 \in \mathcal{A}$ **do**
  $D_o^2 = \sum_{x=1}^{x=k} C_x^2 \times \alpha_x$
end for
function $H\_Dead\_Calc(\mathcal{P}, \mathcal{S})$
  for all $\tau_l \in \mathcal{P} \cup \mathcal{S}$ **do**
    $\Delta_{i_1 j_1} = \sum_{\tau_l \in \mathcal{P} \cup \mathcal{S}} (C_l \times \beta_l^{i_1 j_1})$
    if $\Delta_{i_1 j_1} > r_{i_1 j_1}$ **then**
      $d_{i_1 j_1} = \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{P_{i_1}}{P^s} \rceil) + C_{i_1} + \Delta_{i_1 j_1} - r_{i_1 j_1}$
    else
      $d_{i_1 j_1} = \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{P_{i_1}}{P^s} \rceil) + C_{i_1}$
    end if
  end for
  $D_{i_1} = max\{d_{i_1 j_1}\}$
end function

5 Implementation

5.1 Case Study

We present in this section a simple example of an electric oven whose temperature we want to keep constant after in interruption that may disturb the temperature stability. For example, we want to keep it at 180 °C after a sudden opening of the oven's door as presented in Fig. 5. The oven is heated by an electrical resistance, the intensity of which can vary. Inside the oven there is also a temperature probe, which allows to measure and monitor the temperature in the oven.

This system is implemented by three sets: $\mathcal{P} = \{\tau_1^0, \tau_2^0\}$, $\mathcal{S} = \{\tau_1^1\}$ and $\mathcal{A} = \{\tau_1^2\}$. The tasks are presented in Table 1:

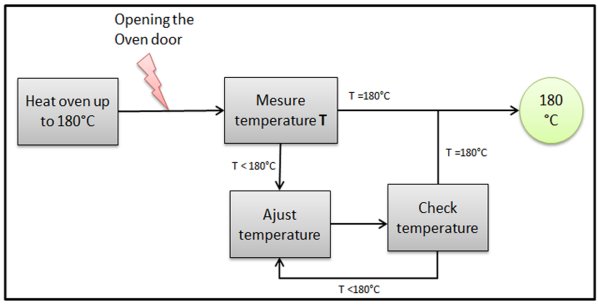

Fig. 5. Electric oven modelisation.

Table 1. System tasks.

Task	Fonction	WCET	Period	Degree of criticality
τ_1^0	Mesures temperature	2	8	1
τ_2^0	Heats the oven	2	16	3
τ_1^1	Checks temperature value	2	16	2
τ_1^2	Adjusts temperature	2		5

We have, $HP = LCM\{8, 16\} = 16s$.

Let's suppose that the parameter λ of the Poisson distribution is equal to 0.5 occurrences in 10 s. Thus, in the hyper-period we have $\frac{HP}{10} \times \lambda = \frac{16}{10} \times 0.5 = 1.6$, i.e., $OC = 2$ occurrences.

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 Eq. (6), $P^s = \lfloor \frac{16}{2} \rfloor = 8$

According to Eq. (8), $Q = 2 \times 2 + 2 \times 1 = 6$

According to Eq. (7), $C^s = \lfloor \frac{16-6}{2} \rfloor = 5$

After that, we calculate the soft deadline of the aperiodic task τ_1^2. According to Eq. (9)

$$D_1^2 = C_1^2 \times \alpha_1 = 2 \times 1 = 2$$

Second step is the calculation of periodic and sporadic tasks' deadlines. 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, and τ_1^1 becomes τ_3.

As an example, we take the calculation of deadline D_1^0 for task τ_1^0, i.e., D_1 for the task τ_1. The number of jobs of task τ_1 in the hyper-period HP is $\frac{HP}{P_3} = \frac{16}{8} = 2$ jobs.

Job τ_{11}:

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

We have $\lfloor \frac{1\times 8}{16} \rfloor \times 16 < 1 \times 8$, i.e., $0 < 8$, then we conclude that $\beta_2^{11} = 0$. In the same way, we have $\beta_3^{11} = 0$

According to Eq. (11), Δ_{11} is calculated as following

$$\begin{aligned}\Delta_{11} &= (1-1) \times 2 + \lfloor \frac{1 \times 8}{16} \rfloor - 0) \times 2 + \lfloor \frac{1 \times 8}{16} \rfloor - 0) \times 2 \\ &= 0 \times 2 + 0 \times 2 + 0 \times 2 = 0\end{aligned}$$

We have $r_{11} = 0$, so $\Delta_{11} = r_{11}$. Thus, according to Eq. (13),

$$\begin{aligned}d_{11} &= \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_1 \\ &= 2 + 2 = 4\end{aligned}$$

Job τ_{12}:

First of all, we calculate the job quantity Δ_{12}. According to Eq. (11), we have to calculate β_2^{12}, and β_3^{12} as indicated in Eq. (12). We have $\lfloor \frac{2\times 8}{16} \rfloor \times 16 < 2 \times 8$, i.e., $16 = 16$, and $\phi_1 < \phi_2$ then we conclude that $\beta_2^{12} = 1$. In the same way, wa have $\beta_3^{11} = 1$

According to Eq. (11), Δ_{12} is calculated as following

$$\begin{aligned}\Delta_{12} &= (2-1) \times 2 + \lfloor \frac{2 \times 8}{16} \rfloor - 1) \times 2 + \lfloor \frac{2 \times 8}{16} \rfloor - 1) \times 2 \\ &= 1 \times 2 + 0 \times 2 + 0 \times 2 = 2\end{aligned}$$

We have $r_{12} = 8$, then $\Delta_{12} < r_{12}$ and we have

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

Thus, according to Eq. (13),

$$d_{12} = \sum_{\tau_l^2 \in \mathcal{A}} (C_l^2 \times \lceil \frac{Pi_1}{P^s} \rceil) + C_1 = 2 + 2 = 4$$

Finally, we calculate the deadline D_1 of the task τ_1 as bellow

$$D_1 = max\{d_{11}, d_{12}\} = max\{4, 4\} = 4$$

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_1^1	τ_2^2
Calculated deadline	4	10	6	2

Figure 6 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.

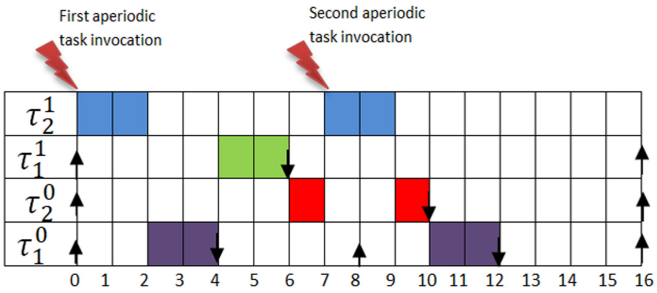

Fig. 6. Scheduling of tasks after the execution of the proposed approach.

5.2 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 [15], where the critical scaling factor (CSF) algorithm is developed.

Figure 7 visualizes simulation that compares the proposed approach with the work reported in [15], where the critical scaling factor (CSF) algorithm is developed. We obtain better results in terms of decrease rate of deadlines in the proposed approach. In fact, the reduction rates of deadlines by using [15] are smaller than those by using the proposed work. The gain is more significant when increasing the number of tasks. If 10 tasks are considered, then the gain

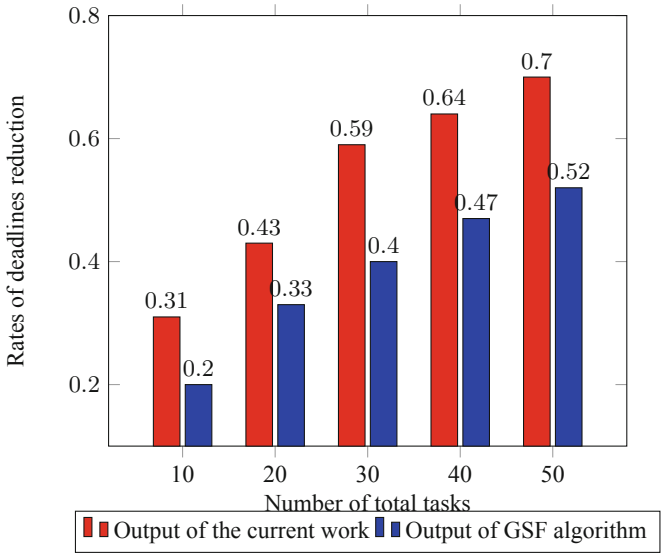

Fig. 7. Rates of deadlines reduction in the case of the proposed approach and in the case of GSF algorithm.

is equal to $(0.31 - 0.2) = 0.11$, and if 50 tasks are considered, then the gain is equal to $(0.7 - 0.52) = 0.18$.

As it was presented in [22], Fig. 8 shows that the NPS algorithm serves to improve the aperiodic response time compared to background service (BK), deferrable server (DS) and total bandwidth server (TBS).

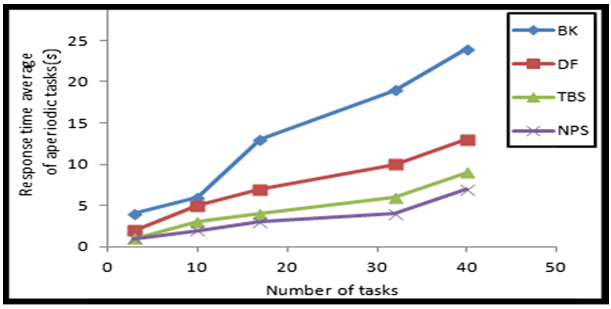

Fig. 8. The improvement of aperiodic tasks response times [22].

5.3 Comparative Study

Table 3 describes the comparison of the developed approach in this paper with some studies. The originality is manifested by treating different and independent

Table 3. Comparative study.

Work	Deadline calculation	Tasks' type	Offline/Online
[11]	This work considers periodic tasks only	It aims to ensure the system schedulability by managing the tasks of by modifying either their periods or worst-case execution time without considering the deadlines analysis. This orientation affects the performance of the system	Online: all the calculations are done online which can be expensive in case of errors
[18]	Same as [11]	It calculates new deadlines to improve the responsiveness in the context of TBS	Same as [11]
[21]	This work addresses the problem of mixed tasks	It aims to schedule mixed tasks while reducing energy consumption	Same as [11]
The proposed approach	The main objective of this work is to calculate deadlines which guarantee i) the respect of hard real-time constraints for periodic and sporadic tasks, and ii) the improvement of aperiodic tasks response time	It deals with real-time tasks of various types: periodic, sporadic and aperiodic	Offline: which is suitable for the design phase and subsequently it is not expensive in case of errors

problems together, i.e., periodic, sporadic and aperiodic tasks and hard and soft real-time constraints tasks simultaneously.

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.

6 Conclusion

This paper is interested in real-time systems executing periodic, sporadic and aperiodic tasks. Our study concerns specifically the computing off effective tasks' deadlines. We propose a new approach that consists on creating the NPS server, it is a service task invoked periodically to execute aperiodic tasks after having calculated aperiodic tasks' soft deadlines. Then, this approach calculates the periodic and sporadic tasks deadlines based on the degree of criticality of tasks and while considering the invocation of aperiodic task execution. An application to a case study and performance evaluation show the effectiveness of the proposed approach and that the NPS can provide a substantial reduction in the average response time of the aperiodic tasks. In our future works, we will be interested in the implementation of the paper's contribution that will be evaluated by assuming real case studies.

References

1. Lakdhar, Z., Mzid, R., Khalgui, K., Li, Z., Frey, G., Al-Ahmari, A.: Multiobjective optimization approach for a portable development of reconfigurable real-time systems: from specification to implementation. IEEE Trans. Syst. Man Cybern. Syst. **49**(3), 623–637 (2018)
2. Anastasia, M., Jarvis, S., Todd, M.: Real-time dynamic-mode scheduling using single-integration hybrid optimization. IEEE Trans. Autom. Sci. Eng. **13**(3), 1385–1398 (2016)
3. Burns, A., Wellings, A.: Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX, 4th edn. Addison- Wesley Educational Publishers Inc., Boston (2009)
4. Ben Meskina, S., Doggaz, N., Khalgui, M., Li, Z.: Reconfiguration-based methodology for improving recovery performance of faults in smart grids. J. Inf. Sci. **454–455**, 73–95 (2018)
5. Goubaa, A., Khalgui, M., Li, Z., Frey, G., Zhou, M.: Scheduling periodic and aperiodic tasks with time, energy harvesting and precedence constraints on multicore systems. J. Inf. Sci. **520**, 86–104 (2020)
6. Ghribi, I., Ben Abdallah, R., Khalgui, M., Li, Z., Alnowibet, K., Platzne, M.: R-codesign: codesign methodology for real-time reconfigurable embedded systems under energy constraints. IEEE Access **6**, 14078–14092 (2018)
7. Liu, C., Layland, J.: Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM (JACM) **201**, 46–61 (1973)

8. Baruah, S., Goossens, J.: Scheduling real-time tasks: algorithms and complexity. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis, vol. 3 (2004)
9. Von der Brüggen, G., Huang, W., Chen, J., Liu, C.: Uniprocessor scheduling strategies for self-suspending task systems. In: 24th International Conference on Real-Time Networks and Systems, pp. 119–128. Association for Computing Machinery, USA (2016)
10. Shanmugasundaram, M., Kumar, R., Kittur, H.: Performance analysis of preemptive based uniprocessor scheduling. Int. J. Electr. Comput. Eng. **6**(4), 1489–1498 (2016)
11. Gammoudi, A., Benzina, A., Khalgui, M., Chillet, D.: New pack oriented solutions for energy-aware feasible adaptive real-time systems. In: Fujita, H., Guizzi, G. (eds.) SoMeT 2015. CCIS, vol. 532, pp. 73–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22689-7_6
12. Gammoudi, A., Benzina, A., Khalgui, M., Chillet, D., Goubaa, A.: ReConf-pack: a simulator for reconfigurable battery-powered real-time systems. In: Proceedings of European Simulation and Modelling Conference (ESM), Spain, pp. 225–232 (2016)
13. Gasmi, M., Mosbahi, O., Khalgui, M., Gomes, L., Li, Z.: R-node: new pipelined approach for an effective reconfigurable wireless sensor node. IEEE Trans. Syst. Man Cybern. Syst. **486**, 892–905 (2016)
14. Wang, X., Li, Z., Wonham, W.: Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control. IEEE Trans. Industr. Inf. **121**, 101–111 (2015)
15. Balbastre, P., Ripoll, I., Crespo, A.: Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. IEEE Trans. Comput. **571**, 96–109 (2007)
16. Wang, X., Khemaissia, I., Khalgui, M., Li, Z., Mosbahi, O., Zhou, M.: Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks. IEEE Trans. Autom. Sci. Eng. **121**, 258–271 (2014)
17. Wang, X., Khemaissia, I., Khalgui, M., Li, Z., Mosbahi, O., Zhou, M.: Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control. IEEE Trans. Industr. Inf. **121**, 101–111 (2015)
18. Cervin, A., Lincoln, B., Eker, J., Arzén, K., Buttazzo, G.: 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, Sweden, pp. 1–9 (2004)
19. Wang, X., Li, Z., Wonham, W.: Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on DES supervisory control. IEEE Trans. Syst. Man Cybern. Syst. **477**, 1082–1098 (2016)
20. Ripoll, I., Ballester-Ripoll, R.: Period selection for minimal hyperperiod in periodic task systems. IEEE Trans. Comput. **629**, 1813–1822 (2012)
21. Yiwen, Z., Haibo, L.: Energy aware mixed tasks scheduling in real-time systems. Sustain. Comput. Inform. Syst. **23**, 38–48 (2019)
22. Goubaa, A., Khalgui, M., Frey, G., Li, Z.: New approach for deadline calculation of periodic, sporadic and aperiodic real-time software tasks. In: Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020), 452–460 (2020). ISBN 978-989-758-443-5
23. Chetto, M.: Optimal scheduling for real-time jobs in energy harvesting computing systems. IEEE Trans. Emerg. Top. Comput. **22**, 122–133 (2014)

24. Sun, Y., Yuan, Z., Liu, Y., Li, X., Wang, Y., Wei, Q., Wang, Y., Narayanan, V., Yang, H.: Maximum energy efficiency tracking circuits for converter-less energy harvesting sensor nodes. IEEE Trans. Circuits Syst. II Express Briefs **646**, 670–674 (2017)
25. Yang, J., Wu, X., Wu, J.: Optimal scheduling of collaborative sensing in energy harvesting sensor networks. IEEE J. Sel. Areas Commun. **333**, 512–523 (2015)
26. Pillai, P., Shin, K.: Real-time dynamic voltage scaling for low-power embedded operating systems. In: Proceedings of the 13th Euromicro Conference on Real-Time Systems, pp. 59–66. ACM, USA (2001)
27. Buttazzo, G.: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, vol. 24. Springer, Boston (2011). https://doi.org/10.1007/978-1-4614-0676-1