Energy and Quality of Control Constraints in Real-Time Scheduling of Synchronous Hybrid Tasks

A. Abbas[*,], M. Loudini[**], W. K. Hidouci[**]**

[*] *Université Akli Mohand Oulhadj de Bouira, Rue Frères Boussendalah, 10000 Bouira, Algeria*
(e-mail: a_abbas@esi.dz)
[**] *Ecole Nationale Supérieure d'Informatique (ESI), Laboratoire de Communication dans les Systèmes Informatiques (LCSI), B.P 68M, 16270 Oued Smar, El Harrach, Algiers, Algeria*
(e-mails: m_loudini@esi.dz, w_hidouci@esi.dz)

Abstract: The embedded systems are characterized by their autonomic functioning whose energy supply is ensured by batteries. Therefore, the reduction of their power consumption and more specifically, the quality of their control becomes the crucial metric optimization in the design of such systems. In this paper, we investigate the integration of precedence, resources sharing and quality of control constraints in the real time scheduling of firm periodic and aperiodic tasks. To study this problem, we start by adapting the analytical conditions of scheduling having been proposed in literature. We also treat the problem of dynamic voltage scaling of the processor in the same context. The community of the researchers have proposed many dynamic voltage-scaling algorithms. However, these algorithms do not consider the precedence, shared resources and quality of control constraints on the scheduling of firm periodic and aperiodic tasks. In this case, we propose two algorithms with the constraints cited above. These algorithms are based on the analytical model adopted in this paper. Experimental results show that the proposed algorithms reduce the energy consumption under earliest deadline first scheduling policy and the stack resource protocol.

Keywords: Embedded systems; Scheduling algorithms; synchronous constraints; hybrid tasks; power management.

1. INTRODUCTION

This paper treats the real-time scheduling problem of synchronous firm periodic and aperiodic tasks, under energy and quality of control constraints in the embedded systems. These systems are characterized by their autonomies of functioning. For this, the reduction of the energy power on one hand and the insurance of the quality of their control on the other hand become crucial optimization metrics in the conception and the realization of these systems. These systems often require periodic real-time tasks, where some are critical deadline (hard tasks). In practice, many systems are considered hard real-time systems. For example, steer-by-wire application (Wilwert et al., 2003), which consist to send signals from the flywheel to the vehicle wheels. This application requires that the conductor ordering arrive at the actuators of the wheels before a critical deadline, beyond which the vehicle may become uncontrollable. Other periodic real-time tasks can miss their deadline and are usually called soft tasks. In this case, we associate to these tasks a mistake tolerance. In other words, the authorized miss deadlines involve only degradations of Quality of Control (QoC) without compromising the correct operation of the system and without endangering its integrity. Many control systems tolerate the occasional temporal constraint violation. However, the notion of soft real-time system is not appropriate for the majority of these systems in which the manner of loss instances need to be explicitly specified. When a system has periodic soft tasks, in which the many consecutive losses instances need to be explicitly specified, it is characterized as a system with firm tasks. An example is the control of an inverted pendulum: task periodically calculates the next action to perform to keep the pendulum in the position of stable equilibrium (an angle of 180°). Miss some samples (deadlines) of this task can be tolerated (the pendulum remains in equilibrium position, but with more vibration) but if several consecutive instances of this task miss their deadlines, the pendulum will fall. These same systems needed aperiodic tasks with no deadlines (for example, the alarm tasks). They are characterized by the fact that they need to be executed at any time, without having time invocation or period. The only known parameter for these tasks is the execution time. However, we must decrease their response times without compromising the execution of the periodic tasks. Moreover, such systems generally need tasks which are dependent and collaborative to realize the expected objectives. This collaboration is characterized by data exchange and/or resource sharing but it generates constraints of precedence and resource sharing which must be considered in the tasks scheduling analysis. In addition, these systems are characterized by their autonomous functioning whose energy supply is ensured by batteries. Therefore, the reduction of the power consumption became crucial metric. The objective in this case is not only consists in determining the order of execution of the tasks under time constraints and synchronization, but also to fix the frequency of the processor and the supply voltage.

The considered real-time system is composed by periodic and aperiodic tasks. The tasks were scheduled in a system with single processor that supports variable frequency and voltage levels. These tasks are synchronized by the precedence induced by the communications and/or by the access-shared resources in a mutually exclusive manner by the use of semaphores. In this paper, we call a system with periodic and aperiodic tasks a hybrid tasks system.

The outline of the paper is as follows. In section 2, we summarize the related works. The concepts relating to scheduling of the tasks, the constraints of synchronization, quality of control will detailed in section 3. The section 4 describes processor energy consumption model and dynamic voltage scaling. Our main contribution concerning the scheduling of the synchronous hybrid tasks under quality of control will presented in section 5. In section 6, we give two DVS algorithms to compute static slowdown factors in the presence of task synchronization and quality of control to minimize the energy consumption of the system. Experimental results are included and discussed in Section 7. The main conclusions and some future directions are highlighted in section 8.

2. RELATED WORK

In the literature, many algorithms were proposed in order to treat the problem of real-time scheduling of the tasks. For example, in (Liu et al., 1973) the authors found important results for the schedulability of periodic task sets, but under the assumption that tasks are independent. The authors have proposed dynamic priority scheduling algorithm called Earliest Deadline First (noted EDF) and static priority scheduling algorithm called rate monotonic (noted RM). We use, in this paper EDF policy, for which the most urgent task is the one whose absolute deadline is the nearest. When some periodic tasks can miss some consecutive deadlines without compromising the good working of the system, we distinguish the works (Hamdaoui et al., 1995) that proposed the model $(m,k)-\text{firm}$. In (Blazewicz, 1976) the author proposes an approach to treat the precedence constraint under EDF policy. As well as the constraints of precedence, the use of resources generates difficulties related to the access protection to the resources hardware (e.g. memory, network, sensors or actuators) or logical (semaphores or message queues). However, a critical resource cannot be used simultaneously by several tasks and only the task that use this critical resource must be able to go at the end of it use even if it is a lower priority. However, the competition of multiple tasks to access a resource generates delays. In addition, the sharing can cause problems of priority inversion or deadlock. To palliate these problems, resource access protocols were proposed. We mention for example the Priority Inheritance Protocol (noted PIP) and Priority ceiling protocol (noted PCP) given in (Sha et al., 1990) used with RM policy. When EDF is used, the Stack Resource Protocol (noted SRP) proposed in (Baker, 1991). To schedule the aperiodic tasks, two solutions are used. The first is the background treatment. The second is the use of task server. Among this last solution, we evoke the Total Bandwidth Server (noted TBS) proposed in (Spuri et al., 1994). The TBS solution is combined with protocol SRP in (Caccamo et al., 1999; Lipari et al., 2000) in order to schedule the periodic and aperiodic tasks which access to critical resources. The Dynamic Voltage Scaling (noted DVS) is one of the most effective approaches in reducing the power consumption of real-time systems (Nasro et al., 2007). When the required performance of the target system is lower than the maximum performance, supply voltage can dynamically reduced to the lowest possible extent that ensures a proper operation of the system. Recently, many voltage-scheduling algorithms have proposed for hard real-time systems. Thus, when EDF policy is used and in the case of attribution speed to jobs tasks, a polynomial algorithm presented in (Yao et al., 1995), to minimize the power consumption with the scheduling of periodic tasks. In the case of attribution a single speed to all the tasks, the authors of (Aydin et al., 2001) gave an approach based on the necessary and sufficient condition suggested in (Liu et al., 1973). When the tasks share critical resources, (Jejurikar et al., 2002) presents an algorithm to calculate the slowdown factors of the processor. The authors use the analytical condition of scheduling proposed in (Baker, 1991). In (Shin et al., 2004) suggested an algorithm to computation slowdown factors speed of the processor for a system with periodic and aperiodic tasks. Thus, many developed algorithms for DVS approach in real-time systems to scheduling the periodic tasks with different constraints. However, these algorithms do not consider the constraints of sharing critical resources and precedence in scheduling of firm periodic and aperiodic tasks under quality of control. In a recent work (Abbas et al., 2009), we addressed the problem of scheduling real-time tasks in hybrid synchronous energy constraint, which we will aim to deepen and improve in the present work.

3. SYSTEM MODEL

This section describes the system model. We suppose that we know at the beginning the value of the tasks temporal parameters and we suppose that the environment of the system is static.

We consider a single processor system composed by a set of n periodic tasks noted $T=\{t_1,...,t_n\}$ and by a set of h aperiodic tasks noted $Ta=\{ta_1,...,ta_h\}$.

3.1 Periodic tasks

The periodic tasks are modeled by the 6-tuple $t_i=\{r_{i,0},C_i,D_i,P_i,m_i,k_i\}$ and are illustrated by Fig. 1. These parameters are given by the following notations:

- $r_{i,0}$ is the first request time of the task t_i; every release of a task is named a job $t_{i,k}$, $k\geq 0$;
- C_i is the worst case execution time (noted WCET);
- D_i is the relative deadline its k^{th} absolute deadline is $d_{i,k}=r_{i,k}+D_i$;
- P_i is the period of the task with $D_i\leq P_i$

- m_i is the number of deadlines that the task t_i must respect on k_i consecutive activations.

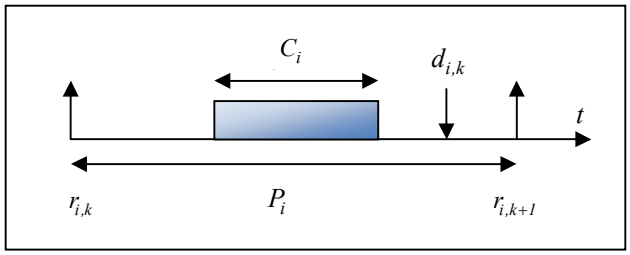

Fig. 1. Execution pattern of real time task.

Each invocation of the task called a job and the k^{th} job of task t_i denoted as $t_{i,k}$ This model offers a general framework to include the hard real-time tasks (when $m = k$) and the firm real time tasks (when $m < k$) In this case, the necessary condition for the feasibility of any schedule under EDF policy (Hamdaoui et al., 1995) is:

$$U = \sum_{i=1}^{n} (m_i \cdot C_i) / (P_i \cdot k_i) \leq 1 \quad (1)$$

where U is the processor utilization by periodic tasks set.

3.2 Quality of Control model

According to the model $(m,k)-$firm (Hamdaoui et al., 1995), every periodic task must respect at least m deadlines on a window of k activations (jobs) consecutive. A mistake occurs, which implies the QoC constraint violation and the not scheduling of system so for k consecutive task jobs, more than $(k-m)$ jobs pass their deadlines. Considering the fact that one of the jobs of the task will not be able to respect its deadline, this it rejected by the system. This politics permits to reduce the system load. In this case, a metric noted T_{QoC} is associated to the QoC constraint that represents the number of jobs of a task that satisfy their deadlines in relation to the number of requests.

$$T_{QoC} = \text{number of satisfied deadlines/number of requests} \quad (2)$$

In (Ramanathan, 1999), proposes an approach permitting to determine if the jobs $t_{i,j}$ of a firm task t_i are optional or critical. Thus, a job $t_{i,j}$ considered critical if:

$$j = \left\lfloor \lceil j \cdot m_i / k_i \rceil \cdot k_i / m_i \right\rfloor \quad (3)$$

In the contrary case, the job is considered optional and can be rejected by the system. We notice that the marking technique of the optional and critical jobs only depends of the ratio m_i/k_i , but not of C_i or P_i .

We consider, in the continuation of this paper, that the firm tasks (with $m < k$) are not concerned by synchronization constraints.

3.3 Aperiodic tasks model

As for the aperiodic tasks, they are modeled by a 1-tuple $ta_i = \{Ca_i\}$, where Ca_i is the worst case execution time. The request time is random because the task execution request depends on the evolution context of the controlled process and cannot be known a priori. The only parameter known for these tasks is the execution time. However, we must decrease their response times, without compromising the execution of the periodic tasks. In real-time systems the response time of a job task is defined as the time elapsed between the request (time when job is ready to execute) to the time when it finishes execution. An aperiodic task set is specified by the mean arrival rate λ and the mean service rate μ .

We use in this paper the concept "*hybrid tasks*" to indicate the periodic and aperiodic tasks.

3.4 Resources model

The system has a set of shared resources, which are accessed by the hard periodic and aperiodic tasks in a mutually exclusive manner by using semaphores. When a task has been granted access to a shared resource, it is said to be executing in its critical section. The k^{th} critical section of task t_i or ta_i which uses a R_k resource is represented as $\beta_{i,k}$. We say a task is blocked if the task has to wait for a lower priority task to release a shared resource and the task holding the resource is called the blocking task. Note that the amount of time a task is blocked is referred to as the task blocking time noted B_i . With the specified task information and with given resource access protocol, the maximum blocking time for a task can be computed.

3.5 Precedence model

The majority of real-time applications require communications between the periodic tasks, which introduce precedence constraints. Thus, we say that there is a precedence constraint between the task t_i and the task t_j or t_i precedes t_j , noted $t_i \rightarrow t_j$, if t_j must await the end execution of t_i to begin its own execution. We assume that the periodic tasks have an atomic form (i.e. normal form), in such way the messages waiting by a task are in beginning, and the emission of messages are at the end of the task. We assume also that the precedence constraints between tasks are simple (i.e $t_i \rightarrow t_j \Rightarrow P_i = P_j$).

4. PROCESSOR ENERGY CONSUMPTION MODEL AND DYNAMIC VOLTAGE SCALING

This section describes the energy consumption model and the dynamic voltage scaling technique, this formulating the problem to be addressed in this paper.

4.1 Energy consumption model

A wide range of processors supports variable voltage and frequency levels. Voltage and frequency levels are tightly coupled. A processor is an integrated circuit of the CMOS family; such a circuit answers the following generic equations (Kwon et al., 2005; Nasro et al., 2007):

4.1.1 Instantaneous power and energy consumed

Power consumption (with joules/second or watt for example) can be divided into two types, static and dynamic:

$$P_{CMOS} = P_{stat} + P_{dyn} \tag{4}$$

where, P_{dyn} is power consumption in switching (dynamic) and P_{stat} is the power consumed by a CMOS gate at rest (static).

In CMOS circuits, the dynamic power represents about 80-85% of the power-dissipated. Conventionally, we neglect the static power. The total power dissipated can be expressed by:

$$P_{CMOS} \approx P_{dyn} \tag{5}$$

The total energy consumed (in joules), at the operating time t, is given by the following formula:

$$E = \left(P_{stat} + P_{dyn}\right) \cdot t = E_{stat} + E_{dyn} \approx E_{dyn} \tag{6}$$

4.1.2 Relationship between frequency and supply voltage

The relationship between operating frequency and supply voltage in the CMOS circuits is given by:

$$f \sim (v - v_t)^{\gamma} / v \tag{7}$$

with γ a constant, v_t the threshold voltage. In the model of metal oxide semiconductor field effect transistor (noted MOSFET) classical γ is approximated by two (Aydin et al., 2001). For a threshold voltage sufficiently small relative to the supply voltage, the relationship between frequency and supply voltage becomes $f \sim v$. In DVS based processors voltage needs to be reduced in proportion to the frequency. Hence, voltage can be expressed as linear function in frequency (Elnozahy et al., 2002):

$$v = \lambda \cdot f \tag{8}$$

4.1.3 Relationship between power, energy, frequency and supply voltage

Dynamic power is calculated by:

$$P_{dyn} = C \cdot f \cdot v^2 = C' \cdot f^3 \tag{9}$$

where, C is a constant related to the type of processor, $C' = C \cdot \lambda^2$, f is the operating frequency and v is the supply voltage.

Under the operating frequency f, one task requires n clock cycle's, which will give a execution time of n/f seconds. By referring to (5), (6) and (9), we obtain the consumed energy equal to:

$$E_{dyn} = (n/f) \cdot C \cdot f \cdot v^2 = n \cdot C' \cdot f^2 \tag{10}$$

4.2 Dynamic Voltage Scaling

The continuous increase in performance and functionality of embedded systems (laptops, personal digital assistant, mobile robots, pacemaker or mobile phones) requires the use of electronic components operating at ever-higher frequencies and hence consuming more energy. Among the methods used to reduce energy consumption, we mention that consisting in the increase of the capacity of the battery. However, this requires increasing its weight, volume and cost. Another method called hybrid, its objective is to make collaborate the software and the hardware. The software carries out an adaptation of consumption, as for the hardware, it reduces consumption. Among the approaches using the hybrid method, the Dynamic Voltage Scaling (noted DVS) (Nasro et al., 2007), aims at the dynamic adaptation of processor voltage and its frequency, to the current needs of the application in performance terms. The scheduler in this case does not limit to define the order of tasks to be executed by the processor. It must also define the processor speed. Taking into account this new dimension generalizes the problem of real-time scheduling. The idea is to use information about the characteristics of the tasks and the scheduling policy to derive the processor speed (slowdown factor noted Sf) that minimizes the total energy consumption in accordance with all deadlines and other constraints of tasks.

We assume that the speed can be varied continuously from $S_{\min}$ to the maximum-supported speed $S_{\max}$ (Aydin et al., 2001). We normalize the speed to the maximum speed to have a continuous operating range of $[S_{\min}, S_{\max}]$, where $S_{\min} = f_{\min} / f_{\max}$ and $S_{\max} = f_{\max} / f_{\max} = 1$ with the minimum and maximum frequencies represented by $f_{\min}$ and $f_{\max}$ respectively. The important point to note is that when we perform a slowdown factor, we change both the frequency and supply voltage of the processor with a scaling factor $Sf \in [S_{\min}, 1]$.

For example, if we consider hard periodic tasks (when $m = k$) without synchronization constraints (without shared resources and precedence's constraints) and if the utilization factor U given in (1) is less than 1 and the processor operates with a maximum speed $S_{\max}$, it is possible to reduce the processor speed (so increasing the execution times of tasks) up to a speed (or to frequency f) which gives a utilization factor equal to 1, i.e. simply take $Sf = U$. To formalize this, we give below the CPU utilization before and after perform a slowdown factor.

When $Sf = S_{\max} = 1$ then $U = \sum_{i=1}^{n} C_i/(P_i \cdot Sf) < 1$ and when $Sf = U$ then $\sum_{i=1}^{n} C_i/(P_i \cdot Sf) = 1$.

The current processor frequency noted f expressed in terms of slowdown factor and the maximum frequency is given as follows:

$$f = f_{\max} \cdot Sf \tag{11}$$

4.2.1 *Relationship between supply voltage, power, energy with slowdown factor*

By referring to the above equations, the current supply voltage v, the current power P_{dyn} relative to the current slowdown factor Sf can be written:

- Current supply voltage:

$$V = V_{\max} \cdot Sf \tag{12}$$

- Current power consumption:

$$P_{dyn} = C' \cdot \left(f_{\max} \cdot Sf\right)^3 = P_{dyn\_\max} \cdot Sf^{\,3} \tag{13}$$

- With current energy consumption per one clock cycle is equal to:

$$E_{dyn} = C' \cdot f^{\,2} = C' \cdot \left(f_{\max} \cdot Sf\right)^2 = E_{dyn\_\max} \cdot Sf^{\,2} \tag{14}$$

with $V_{\max}$, $P_{dyn\_\max}$ and $E_{dyn\_\max}$ is the maximum voltage, the maximum power and the maximum energy consumption under the maximum frequency $f_{\max}$.

4.2.2 *Illustrative Example:*

Consider Γ a hard periodic task set which is composed of four tasks: $\Gamma = \{t_i \mid 1 \le i \le 4\}$ with $t_i = \{C_i, P_i\}$.

Let $t_1 = \{0.1, 3\}$, $t_2 = \{1, 4\}$, $t_3 = \{1, 5\}$ and $t_4 = \{1, 10\}$. Consider also that we have a battery with an initial energy equal to $E_0 = 5$ Joules.

If the processor will operate at a maximum speed with a power consumption of $P_{dyn\_\max} = 0.1$ Joule$/s$ $(= 0.1$ Watt$)$, then after one second $(t = 1)$, the remaining energy will be $E(t=1) = (-0.1*1) + 5 =$ 4.90 Joules.

However, if we use $Sf = U = 0,5833$, the energy remaining during one second will be equal to $E(t=1) = -\left[0.1*1*(0,5833)^2\right] + 5 = 4.966$ Joules and the power consumption equal to $P_{dyn} = 0,1*(0,5833)^3 = 0.0198$ Joules/s.

5. SCHEDULING OF SYNCHRONOUS HYBRID TASKS UNDER QUALITY OF CONTROL CONSTRAINTS

We consider in this section the problem of the scheduling of the synchronous hybrid tasks under quality of control. The objective is to have an order of execution of the tasks that guarantee the respect of $(m,k)-$firm constraint, the use of the critical resources in mutual exclusion and the respect of the precedence constraints by the periodic tasks. In addition, to allow as far as possible, to have a better response time by aperiodic tasks those are also using the critical resources in mutual exclusion.

5.1 Precedence constraints

We exploit the approach that proposed in (Blazewicz, 1976) in order to consider the precedence constraints that exist between periodic tasks during the use of EDF policy. This policy is based on assignments of priority following the temporal parameters of tasks. Thus, the idea is to assign to a task, which must precede another, a priority lower than the task preceding it. This approach suggests the modification of the request time and deadline so that the precedence constraints are implicitly respected. These modifications are operated as follows:

- Computation of request time:

$$r_{0,i}^{\,*} = \max\{r_{0,i},\ \max_{t_j \to t_i} \{r_{0,j}^{\,*} + C_j\}\} \tag{15}$$

From the tasks without predecessors and while going down to the tasks of which all the predecessors were treated.

- Computation of deadline:

$$D_i^* = \min\left\{D_i,\ \min_{t_j \to t_i} \{D_j^* - C_j\}\right\} \tag{16}$$

From the tasks without successors and while going up to the tasks of which all the successors were treated.

We note that these modifications are operated in off-line i.e. before the effective scheduling of the tasks. Moreover, we must consider in the continuation that the deadline of a task can be lower than its period, and requests times are different following the modifications operated previously.

5.2 Precedence and resource constraints

We choose to use the SRP protocol (Baker, 1991) to manage resource access in mutual exclusion and to compute the maximum blocking time for a task, with taking into account the constraints precedence. This protocol is based on some principles, which are summarized as follows:

- In addition to its priority, each task is been allotting statically (off-line), a parameter π called preemption level. These levels are assigned in a way inversely proportional to the relative deadline.

- Each resource R_k is been assigning dynamically a ceiling value noted CR_k, determined by the maximum value of the preemption levels of the active tasks needing more than the unit available resource of R_k (i.e., the tasks which demand for units of semaphore cannot be satisfied).

$$CR_k = \max_i \{\pi_i / t_i \ is\ blocked\ at\ R_k\} \tag{17}$$

- The system ceiling noted Π, is the maximum value of the ceilings resources in use.

A task t_i, can preempt task t_j if the following conditions are verified:

- Its absolute deadline d_i is lower than that of t_j;
- The preemption level of t_i is higher than the preemption level of t_j (i.e., $\pi_i > \pi_j$);
- Its preemption level is higher than the system ceiling (i.e $\pi_i > \Pi$).

In order to carry out a scheduling analysis of n periodic synchronous tasks under QoC constraints, we adapt the scheduling sufficient condition established in (Baker, 1991), as follows:

$$\forall i = 1,...,n\ ;\ \sum_{j=1}^{i} \frac{m_j \cdot C_j}{k_j \cdot D_j} + \frac{m_i \cdot B_i}{k_i \cdot D_i} \le 1 \tag{18}$$

B_i represents the maximum blocking time of the task t_i, it is equal to the biggest duration of the critical section of the periodic tasks, which have a lower preemption level than that of tasks t_i and it uses a resource having the ceiling value higher or equal to the preemption level of t_i. To consider the precedence constraints, we suggest that the tasks having lower preemption level considered are those not on relations, directly or indirectly, by constraints of precedence with the task t_i. This duration (i.e. B_i) is given by the following formula:

$$B_i = \max_{j,k} \{\beta_{j,k} / \pi_i > \pi_j \wedge C_{Rk} \ge \pi_i \wedge t_j \notin \{succ\ \ de\ \ t_i\}\} \tag{19}$$

where $\beta_{j,k}$ represents the critical section of task t_j, which have a lower preemption level π_j than that of tasks t_i and which use a resource having the ceiling CR_k value higher or equal to the preemption level of t_i. The task t_j is considered if it is not included in the set of predecessors or successors of t_i.

5.3 Aperiodic tasks

When the system is composed by periodic and aperiodic tasks, which share critical resources, we consider the two following approaches:

5.3.1 First approach

The first approach of the aperiodic tasks scheduling is based on the idea developed in (Caccamo et al., 1999). The authors suggested assigning at each aperiodic request a deadline and preemption level so that they can be scheduled by SRP protocol with the use of the total bandwidth server noted TBS. The attribution of deadline is done as follows:

$$da_k = \max\left(ra_k, da_{k-1}\right) + \frac{Ca_k}{U_s} \tag{20}$$

where ra_k is the invocation date of the k^{th} aperiodic task job, which is known as online, and Ca_k its execution time. da_{k-1} is the deadline of the precedent job (with $da_0 = 0$) and U_s is the load of the used server. This load admits for max value:

$$U_s = 1 - U \tag{21}$$

where U is the periodic tasks processor utilization, which is calculated referring to (1). This attribution of deadline respects the fact that the use of the processor by the aperiodic ones never exceeds the value U_s of the server.

As for the preemption level, it is calculated as follows:

$$\pi = \frac{U_s}{Ca_k} \tag{22}$$

For the analysis of the synchronous hybrid tasks scheduling under quality of control constraints, we improve the sufficient condition, which was proposed in (Lipari et al., 2000) such as the tasks are in ascending order of their preemption levels; the analytical condition is as follows:

$$U + U_s \le 1 \tag{23}$$

$$\forall i, 1 \le i \le n,$$

$$D_i \ge \sum_{j=1}^{i} \left\{ \left\lfloor \frac{D_i - D_j}{P_j} \right\rfloor + 1 \right\} \frac{m_j \cdot C_j}{k_j} + \max\left\{0, \frac{m_i \cdot B_i}{k_i} - 1\right\} + S(i) \cdot D_i \cdot U_s \tag{24}$$

with $S(i)$ is the selection function which takes as values:

$$S(i) = \begin{cases} 0 & if\ \pi_i \ge \pi^* \\ 1 & if\ \pi_i < \pi^* \end{cases} \tag{25}$$

where π^* represents the maximum preemption level of the aperiodic tasks, π_i is the preemption level of the periodic task t_i. D_i is the deadline of the task t_i, D_j and P_j are respectively the deadline and the period of the periodic task t_j having a preemption level lower than that of t_i. m_i is the number of deadlines that the task t_i must respect on k_i

consecutive activations. U is the processor utilization by periodic tasks and U_s is the load of the used server. The parameter B_i represents the maximum blocking time of the task t_i calculated referring to (19).

5.3.2 Second approach

In the second approach, we consider that the aperiodic tasks are managed by a server task, which is the higher priority periodic task, and its execution time is equal to the sum of the execution times of the aperiodic tasks. This server task treats at each activation the aperiodic tasks blocked in the queue, according to policy FIFO (first in first out), until exhaustion of its execution time.

To analyze the scheduling of the synchronous hybrid tasks under QoC constraints, we used a scheduling sufficient condition given above in (18) which we adapt as follows:

$$\forall i = 1,...,n+1;\ \sum_{j=1}^{i} \frac{m_j \cdot C_j}{k_j \cdot D_j} + \frac{m_i \cdot B_i}{k_i \cdot D_i} \le 1 \tag{26}$$

where $n+1$ represents the number of periodic tasks with the task server and B_i the maximum blocking time of the task t_i, which is calculated referring to (19).

6. TASK SLOWDOWN FACTORS COMPUTATION

In this section, we compute task slowdown factors in the presence of hybrid tasks synchronization. We present two algorithms, which used to compute slowdown factors according to the two approaches presented previously.

6.1 Algorithm I

In (Shin et al., 2004) authors gave an algorithm to solve the problem of the determination of the task slowdown factors, to reduce the power consumption during the scheduling of the periodic and aperiodic tasks. This algorithm determines two-slowdown factors in order to determine the speed of execution of the tasks. The first relates to the periodic tasks noted Sp, when the other noted Ss relates to the aperiodic tasks. However, authors do not consider of resources, precedence's constraints and quality of control.

For the determination of the slowdown factors, we based on the model presented in the first approach. We formulate the problem as follows:

Given:

U, U_s, ω, ρ, $\{t_1,\ldots,t_n\}$ of n periodic tasks in increasing order of their deadlines with their characteristics (C_i, D_i, P_i, B_i, m_i, k_i), the maximum preemption level of the aperiodic tasks π^*.

Find:

Sp, Ss which minimize energy E given as

$$E = Sp^2 \cdot \omega + Ss^2 \cdot \rho \tag{27}$$

Subject to:

$$\frac{U_s}{Ss} + \frac{U}{Sp} = 1 \tag{28}$$

$$\forall i, 1 \le i \le n,$$

$$D_i \ge \sum_{j=1}^{i} \left\{ \left\lfloor \frac{D_i - D_j}{P_j} \right\rfloor + 1 \right\} \frac{m_j \cdot C_j}{k_j \cdot Sp} + \max\left\{0, \frac{m_i \cdot B_i}{k_i \cdot Sp} - 1\right\} + \frac{S(i) \cdot D_i \cdot U_s}{Sp} \tag{29}$$

$$0 < Sp,\ Ss < 1 \tag{30}$$

taking into account (23) and (24).

ω is the average workload ratio of periodic tasks ($\omega = U/n$), ρ is the average workload ratio of aperiodic tasks ($\rho = \lambda/\mu$) and Sp and Ss are the two slowdown factors.

We give the solution to this problem by using Lagrange transform as follows:

$$Sp = U + U_s \sqrt[3]{\frac{\rho}{U_s * \omega}} \tag{31}$$

$$Ss = U_s + U \sqrt[3]{\frac{U_s * \omega}{\rho}} \tag{32}$$

6.2 Algorithm II

In this paragraph, we give an algorithm when the second approach is used, which calculates the slowdown factor noted Sf so that the following analytical condition is verified:

$$\forall i = 1,...,n+1;\ \sum_{j=1}^{i} \frac{m_j \cdot C_j}{k_j \cdot D_j \cdot Sf} + \frac{m_i \cdot B_i}{k_i \cdot D_i \cdot Sf} \le 1 \tag{33}$$

Algorithm II. Slowdown factors Sf computation

Given: $\{t_1,\ldots,t_n\}$ n periodic tasks in increasing order of their deadlines with their characteristics (C_i, D_i, P_i, B_i, m_i, k_i)

$X \leftarrow 0,\ Y \leftarrow 0,\ i \leftarrow 1,\ V \leftarrow 0$ // initialisation

while ($i \le n$) do

$j \leftarrow 1$;

while ($j \le i$) do

$X += \dfrac{m_j \cdot C_j}{K_j \cdot D_j}$;

$j \leftarrow j + 1$;

end while

$$Y \leftarrow \frac{m_i \cdot B_i}{k_i \cdot D_i} \ ;$$

if ($X + Y > 1$) then

Exit() ; // sufficient condition not respected

end if

$$V = Max(V, X + Y) \ ;$$

$$X \leftarrow 0 \ ;$$

$$i \leftarrow i + 1 \ ;$$

end while

$Sf \leftarrow V$.

7. EXPERIMENTAL RESULTS

This section describes the carried out simulations and the experimental results in order to evaluate the two approaches presented previously and to compare between the two dynamic voltage scaling (DVS) algorithms developed in the framework of our study. Overall, the simulation results were promising.

7.1 Environment and context of simulation

developed a simulator program with VC++, in which we implemented the dynamic policy EDF, protocol SRP and algorithms that we had developed. To carry out simulations, we randomly generated some Task-sets. For that, we generated according to the uniform law characteristics concerning the periodic tasks.

Periods (P_i) are generated in the interval [50, 80], the worst-case execution times (C_i) in the interval [4, 8]. Resources R_i from 0 to 2 and the position of the critical section are randomly generated. The length of the critical section ($\beta_{i,k}$) of the task t_i that access to R_k is generated in the interval [1, C_i]. The precedence of the task t_i with other tasks (from 0 to $n-1$). The parameters m_i and k_i of the $(m,k)-$ firm model are randomly generated. The k_i are uniformly distributed in the range [4, 10] and m_i in the interval [2, k_i].

In the case of simulations, the instance of a task is rejected by the system if it is listed as optional by the model given by (3). For the aperiodic tasks, the interval of time between two arrivals is a Poisson process with parameter $\lambda = 0.01$ and the execution time is exponential process with parameter $\mu = 0.25$. Such as $1/\lambda$ is the average duration of the inter-arrivals, $1/\mu$ is the average duration of service. Theoretically, response time of the aperiodic tasks is calculated by the M/M/1 Markov model as $(\mu - \lambda)^{-1} = 4.16$.

We assume that the system is powered by a battery with an initial charge of $E_0 = 172800$ Joules $(48$ Wh$)$ with a power consumption of $P = 75$ Watts under maximum operating frequency ($f_{\max}$) and the system is simulated for a duration of 2000 milliseconds.

7.2 Results and discussions

According to the context specified previously, we randomly generated six sets from 2 to 7 periodic tasks with one aperiodic task in each set. Then, these sets were used to test the two approaches and the two algorithms. Note that in the first approach case, we assign to TBS a load equal to $U_s = 0.25$ and in the second one, we also generate a task server having a high priority (with the smallest deadline). In what follows, we give and discuss the simulation results illustrated by a set of tables and figures.

Table 1. Results of using the algorithm I (first approach) before and after calculates and use the slowdown factor.

Task Sets	Algorithm I (*Us* =0.25)								
	Before			Slowdown factors		After			
	U	*Tqoc (%)*	*Average response time*	*Sp*	*Ss*	*Tqoc (%)*	*Average response time*	*Remaining energy with DVS (Wh)*	*Energy profit (%)*
1	0.211	24.31	4.00	0.371	0.647	24.65	10.79	42,28	74.88
2	0.289	24.31	4.00	0.434	0.682	24.65	9.22	40,16	70.47
3	0.384	24.65	4.00	0.519	0.741	24.65	7.71	36,79	63.45
4	0.466	18.86	4.00	0.598	0.801	18.86	6.69	33,10	55.76
5	0.587	18.86	4.00	0.723	0.897	18.86	5.53	26,21	41.42
6	0.707	18.86	4.00	0.856	0.998	18.86	4.67	17,45	23.15

Table 2. Results of using the algorithm II (second approach) before and after calculates and use the slowdown factor.

Task Sets	Algorithm II							
	Before			Slowdown factor	After			
	U	*Tqoc (%)*	*Average response time*	*Sf*	*Tqoc (%)*	*Average response time*	*Remaining energy with DVS (Wh)*	*Energy profit (%)*
1	0.291	24.65	23.21	0.291	24.65	29.60	44,48	79.47
2	0.369	13.92	23.21	0.369	13.92	30.82	42,33	75.00
3	0.470	12.12	23.21	0.470	12.12	31.47	38,79	67.62
4	0.607	9.77	23.21	0.607	9.77	24.12	32,63	54.78
5	0.684	9.00	20.84	0.684	9.00	21.21	28,49	46.15
6	0.771	9.00	20.89	0.771	9.00	20.19	23,21	35.16

By comparing the results presented in Table 1 and Table 2, we can see that algorithm I (first approach) gives the best average response time before and after we perform a slowdown factor *Sf*. However, the algorithm II (i.e. the second approach) allows less energy consumption (energy remaining and the energy profit) and also a quality of control. This is due to the fact that we have allotted a raised load to the server TBS ($U_s = 0.25$).

Moreover, when we allot a weak load to TBS ($U_s = 0.01$), as shown in Table 3, we note that the average response time is significantly higher than when use the algorithm II (second approach). This is due to the assignment of a higher deadline to aperiodic task (refer to (20)) which makes it the lowest priority under EDF policy.

In terms of energy remaining following the use of DVS technique in the first approach with the rest which is equal to 6.33 Wh without DVS, the first approach provides a fair, gain and less consumption of energy. This is due to the time left free by the use of $U_s = 0.01$ which makes the processor idle.

Table 3. Results of using the algorithm I (first approach) before and after calculates and use the slowdown factor.

Task Sets	Algorithm I (*Us*=0.01)								
	Before			Slowdown factors		After			
	U	*Tqoc (%)*	*Average response time*	*Ss*	*Sp*	*Tqoc (%)*	*Average response time*	*Remaining energy with DVS (Wh)*	*Energy profit (%)*
1	0.211	24.31	5.30	0.051	0.262	24.65	379.96	47,89	86.58
2	0.289	24.31	6.20	0.073	0.335	24.65	380.60	47,78	86.34
3	0.384	24.65	7.50	0.102	0.426	24.65	379.83	47,57	85.90
4	0.466	18.86	9.90	0.129	0.505	18.86	377.31	47,31	85.36
5	0.587	18.86	12.95	0.172	0.623	18.86	371.53	46,77	84.24
6	0.707	18.86	17.35	0.217	0.741	18.86	367.96	46,03	82.70

The following graph (Fig. 22) shows the variation of the average response time before we use the DVS (i.e. before performing the processor with the slowdown factors) according to the loads allotted to the TBS server (U_s) during the scheduling of seven periodic tasks and one aperiodic task in the case of using algorithm I (first approach).

From Fig. 22, we can say that when we allot a low load to TBS, the average response time is higher than the theoretical value which is equal to $(\mu - \lambda)^{-1} = 4.16$. This is due to the lowest priority of aperiodic task under EDF policy with referring to (20). For TBS allowance greater than or equal to 0.1, the average response time is very close to the theoretical value. We can also say that for these loads, the aperiodic task is considered of the highest priority.

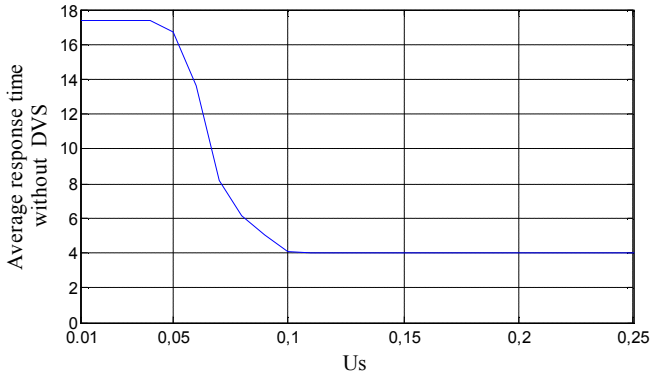

Fig. 2. Variation of average response time according to the variation of U_s without DVS.

Based on the graph given in Fig. 3 illustrating the variation of the average response time when we use the DVS, we can say that the average response time is very large compared to the theoretical value, when the load is allocated to the TBS less than 0.1. However, when U_s takes values greater than 0.2, the average response time is equal to the theoretical value.

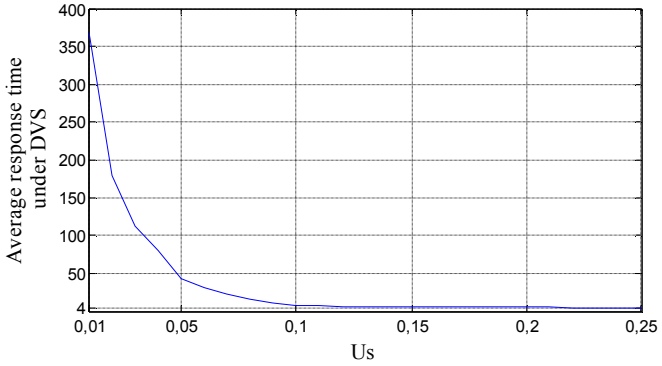

Fig. 3. Variation of the average response time according to the variation of U_s under DVS.

The graph, given hereafter in Fig. 4, shows the variation of the energy profit, under the variation of loads allotted to the TBS server when using algorithm I (first approach) with scheduling of seven periodic tasks and an aperiodic one.

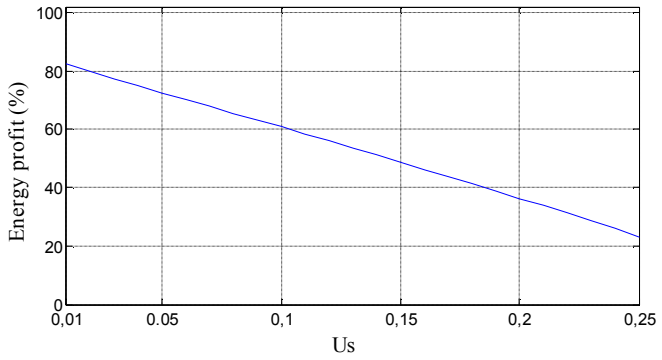

Fig. 4. Variation of the energy profit according to the variation of U_s under DVS.

The Fig. 4 indicates that when we increase the load allocated to the server TBS, the energy profit decreases. This is due to the fact that when a small load is allocated, it leaves the processor idle for a long time. This idle time is used by DVS to reduce energy consumption. However, if a large load is allocated to the server TBS, it will let the processor idle for a little time which is not profitable for the DVS.

With the aim of showing the variation of average response time and energy profit according to a load allotted to server TBS, we normalized the average response times by the maximum value, the result is illustrated by the graph shown in Fig. 5.

Based on this graph, we can say that the increased allocated load to the server TBS decreases the gain in energy and the average response time. Therefore, a compromise should be made between reducing energy consumption and reducing response time.

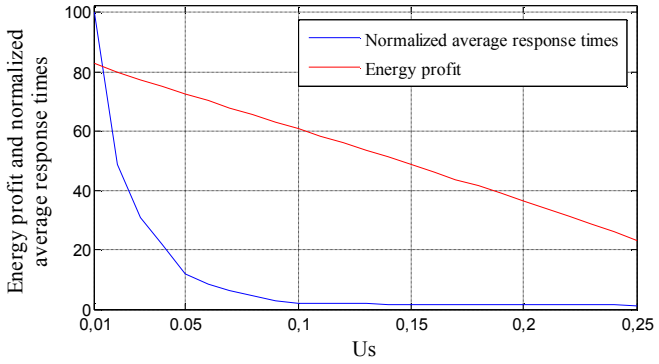

Fig. 5. Variation of the normalized the average response times and the energy profit according to the variation of U_s .

8. CONCLUSIONS AND FUTUR WORKS

In this article, we proposed two algorithms for the dynamic voltage scaling in a context of real-time scheduling of synchronous hybrid tasks constraint. Constraints of sharing resource, precedence and quality of control were studied in a context of energy minimization. We showed that the two algorithms gave convincing results in terms of simulated performances. However, in a real implementation of the algorithms a compromise should be made between the gain of energy and the average response time of the aperiodic tasks. According to the experimentations carried out, we can say that our approaches are promising.

In the future, we plan to combine the use of DVS and energy harvesting with the capabilities of a feedback scheduler in order to take into consideration the variation in execution time of real time tasks.

REFERENCES

Abbas, A., Hentous, H. and Kenaza, T. (2009). Scheduling Real-Time of the Synchronous Hybrid Tasks under Energy Constraint. In *Proc. of the 3rd International Conference on Sensor Technologies and Applications, (SENSORCOMM'09)*, Athens, Glyfada, Greece, 240-245.

Aydin, H., Melhem, R., Mossé, D., and Mejia-Alvarez, P. (2001). Determining Optimal Processor Speeds for Periodic Real-Time Tasks with Different Power Characteristics. In *Proc. of the 13th Euromicro Conference on Real-Time Systems*, Delft, The Netherlands, 225-232.

Baker, T.P. (1991). Stack-Based Scheduling of Real-Time Processes. *Real-Time Systems*, 3(1), 67–99.

Blazewicz, J. (1976). Scheduling Dependent Tasks with Different Arrival Times to Meet Deadlines. In *Proc. of the International Workshop organized by the Commision of the European Communities on Modelling and Performance Evaluation of Computer Systems*, Amsterdam, The Netherlands, 57-65.

Caccamo, M., Lipari G., and Buttazzo, G. (1999). Sharing Resources among Periodic and Aperiodic Tasks with Dynamic Deadlines, In *Proc. of the 20th IEEE Real-Time Systems Symposium*, Phoenix, AR, USA, 284-293.

Elnozahy, E. N., Kistler, M., and Rajamony R. (2002). Energy-efficient server clusters. *In Proc. of the 2nd International Conference on Power-Aware Computer Systems (PACS'02)*, 179-197.

Hamdaoui, M., and Ramanathan, P. (1995). A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines. *IEEE Transactions on Computers*, 44(12), 1443–1451.

Jejurikar, R. and Gupta, R. (2002). Energy Aware Task Scheduling with Task Synchronization for Embedded Real Time Systems. In *Proc. of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, Grenoble, France, 164-169.

Kwon, W.C., and Kim, T. (2005) Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors. *ACM Transactions on Embedded Computing Systems*, 4(1), 211-230.

Lipari, G. and Buttazzo, G. (2000). Schedulability Analysis of Periodic and Aperiodic Tasks with Resource Constraints. *Journal of Systems Architecture*, 46(4), 327-338.

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

Min-Allah, N., Wang, Y.-J., Xing, J.-S., Nisar, W. and Kazmi, A.-R. (2007). Towards Dynamic Voltage Scaling in Real-Time Systems – A Survey. *International Journal of Computer Sciences and Engineering Systems*, 1(2), 93-103.

Ramanathan, P.A. (1999). Overload Management in Real-Time Control Applications using (m,k)-Firm Guarantee. *IEEE Trans. on Parallel and Distributed Systems*, 10(6), 549–559.

Sha, L., Rajkumar, R.R and Lehoczky, J.P. (1990). Priority Inheritance Protocols: an Approach to Real-Time Synchronization, *IEEE Transactions on Computers*, 39(9), 1175-1185,

Shin, D., and Kim, J. (2004). Dynamic Voltage Scaling of Periodic and Aperiodic Tasks in Priority-Driven Systems. In *Proc. of the 2004 Asia and South Pacific Design Automation Conference (ASP-DAC'04)*, Piscataway, NJ, USA, 653-658.

Spuri, M. and Buttazzo, G.C. (1994). Efficient Aperiodic Control under Earliest Deadline Scheduling. In *Proc. of the IEEE Real-Time Systems Symposium*, San Juan, Portorico.

Wilwert, C., Song, Y.Q., Simonot-Lion, F., and Clément, T. (2003). Evaluating Quality of Service and Behavioral Reliability of Steer-by-wire Systems. In *Proc. of the 9th IEEE International Conference on Emerging Technologies and Factory Automation*, Lisbon, Portugal, 193-200.

Yao, F., Demers, A.J., and Shenker, S.J. (1995). A Scheduling Model for Reduced CPU Energy. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, USA, 374-382.