The Fixed Priority Scheduling Problem for Energy Harvesting Real-Time Systems

Younès Chandarli, Yasmina Abdeddaïm, Damien Masson

To cite this version:


HAL Id: hal-00724622
https://hal-upec-upem.archives-ouvertes.fr/hal-00724622
Submitted on 22 Aug 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
The Fixed Priority Scheduling Problem for Energy Harvesting Real-Time Systems

Younès Chandarli∗†, Yasmina Abdeddaïm∗, and Damien Masson∗
Université Paris-Est, LIGM UMR CNRS 8049,
∗ESIEE Paris, 2 bld Blaise Pascal, BP 99, 93162 Noisy-le-Grand CEDEX, France
†Université Paris-Est Marne-la-vallée, 5 bld Descartes, 77454 Marne-la-Vallée Cedex 2, France

Abstract—Energy harvesting is the process of generating electrical energy from environmental sources such as solar panels. In recent years, this term has been frequently applied in the context of small autonomous devices such as wireless sensor nodes. The classical scheduling theory is insufficient for this kind of systems and new scheduling problems arise in this context. Until now, the research on this area focused in trying to improve the efficiency of existing algorithms. Our approach is to complete these efforts by a feasibility theory allowing us to understand why classical optimal algorithms are not efficient anymore with energy constraints.

In this paper, we try to establish a schedulability test for a fixed priority real-time scheduling problem with energy constraints. We first introduce the problem and describe the model. Then, to illustrate the difficulty of the problem, we focus on a preemptive fixed priority scheduling policy where all the executions are postponed as long as possible. This policy lets the harvester the maximal amount of time to refill the battery. We call this policy PFP\textit{ALAP} for As Late As Possible. We try to define sufficient and/or necessary schedulability conditions and discuss its potential optimality under some additional assumptions. Then, through simple counter examples, we show that intuitive assumptions are wrong for this scheduling problem, making it very interesting to study.

I. INTRODUCTION AND RELATED WORK

The first work addressing the scheduling problem for energy harvesting systems was the one of Mossé [1]. The problem was solved under a very restrictive task model: the frame based model where all the tasks have exactly the same period and implicit deadline. Then Moser et al. studied a very similar problem [2]. They proposed an optimal algorithm called LSA but in their hypotheses, the CPU frequency can be changed to adjust the Worst Case Execution Time (WCET) of the tasks depending on their energy consumption. Then the results of this work rely on a strong hypothesis: tasks energy consumption is directly linked to their WCET. Recent work shows that this hypothesis is not realistic [3]. Finally, the problem was addressed by Chetto et al. in [4, 5]. A clairvoyant algorithm and several non clairvoyant heuristics was proposed. This clairvoyant algorithm relies on a meta policy: as long as the system can perform without energy failure, a standard policy such as EDF is used. Then, as soon as a future energy issue is detected, i.e $E < 0$, the system is paused as long as possible or until the energy storage unit is full. The notion of slack time [6] is extended to the notion of slack energy. An algorithm to compute slack energy under EDF is provided.

However, as most embedded systems only use fixed priority, we are interested in extending and completing existing works for fixed priority scheduling.

The remaining of the paper is organized as follow, we present the model in Section [II] then we describe the PFP\textit{ALAP} algorithm and we discuss about its worst case scenario, its study period and we try to build a schedulability condition based on PFP\textit{ALAP} in Section [III]. Section [IV] presents the problem of finding the minimal battery size that permits to schedule a system. Finally we conclude in Section [V].

II. MODEL

A. Target application description

We consider an embedded system connected to an energy harvesting device. An energy harvesting device is a system that collects energy from its environment. The energy is stored in an energy storage unit. We suppose that we can know or bound the energy quantity that arrives in the storage unit, and for each task the energy that it consumes in the worst case.

Figure 1 illustrates this description.

![Energy Harvesting Embedded System Model](image)

However, as most embedded systems only use fixed priority, we are interested in extending and completing existing works for fixed priority scheduling.

The remaining of the paper is organized as follow, we present the model in Section [II] then we describe the PFP\textit{ALAP} algorithm and we discuss about its worst case scenario, its study period and we try to build a schedulability condition based on PFP\textit{ALAP} in Section [III]. Section [IV] presents the problem of finding the minimal battery size that permits to schedule a system. Finally we conclude in Section [V].

II. MODEL

A. Target application description

We consider an embedded system connected to an energy harvesting device. An energy harvesting device is a system that collects energy from its environment. The energy is stored in an energy storage unit. We suppose that we can know or bound the energy quantity that arrives in the storage unit, and for each task the energy that it consumes in the worst case.

Figure 1 illustrates this description.

Figure 1. Energy Harvesting Embedded System Model

However, as most embedded systems only use fixed priority, we are interested in extending and completing existing works for fixed priority scheduling.

The remaining of the paper is organized as follow, we present the model in Section [II] then we describe the PFP\textit{ALAP} algorithm and we discuss about its worst case scenario, its study period and we try to build a schedulability condition based on PFP\textit{ALAP} in Section [III]. Section [IV] presents the problem of finding the minimal battery size that permits to schedule a system. Finally we conclude in Section [V].

II. MODEL

A. Target application description

We consider an embedded system connected to an energy harvesting device. An energy harvesting device is a system that collects energy from its environment. The energy is stored in an energy storage unit. We suppose that we can know or bound the energy quantity that arrives in the storage unit, and for each task the energy that it consumes in the worst case.

Figure 1 illustrates this description.

Figure 1. Energy Harvesting Embedded System Model

However, as most embedded systems only use fixed priority, we are interested in extending and completing existing works for fixed priority scheduling.

The remaining of the paper is organized as follow, we present the model in Section [II] then we describe the PFP\textit{ALAP} algorithm and we discuss about its worst case scenario, its study period and we try to build a schedulability condition based on PFP\textit{ALAP} in Section [III]. Section [IV] presents the problem of finding the minimal battery size that permits to schedule a system. Finally we conclude in Section [V].
the energy consumption $E_i$ of a task are fully independent. A task $\tau_i$ releases an infinite number of jobs separated by $T_i$ time unit and each job must execute during $C_i$ and consume $E_i$ energy unit. Deadlines are constrained or implicit. The system is powered by a battery or a capacitor recharged from a renewable energy source like a solar panel. The energy is replenished continuously even during the execution of tasks and the energy level of the battery fluctuate between two thresholds $E_{\text{max}}$ and $E_{\text{min}}$. We note $Pr(t)$ the charging function and for the sake of simplicity we consider that $P_r(t)$ is a linear function. We define the processor utilization of $\tau_i$ as $U_i = C_i/T_i$ and its energy utilization as

$$U_{\epsilon_i} = \frac{C_i}{\int_{t}^{t+T_i} Pr(t) \, dt}.$$ 

The total utilization of the system is the sum of all the tasks utilization; i.e $U = \sum U_i$ and $U_{\epsilon} = \sum U_{\epsilon_i}$.

III. **PFP\textsubscript{ALAP} Scheduling Algorithm**

**PFP\textsubscript{ALAP}** is a fixed-priority scheduling policy that delays the jobs executions as long as possible, i.e, as long as the slack time remains positive. We are interested in studying this policy since this algorithm is simple and seems optimal when slack time remains positive. We are interested in studying this algorithm since the jobs executions as long as possible, i.e, as long as the slack-time is available. So without energy constraints, the study period of a task set is still open.

A. **PFP\textsubscript{ALAP} worst case scenario**

For the classical work conserving fixed priority algorithm, which we denote **PFP\textsubscript{ASAP}** by opposition to **PFP\textsubscript{ALAP}**, the worst case activation scenario corresponds to the synchronous activation of all tasks [7]. In this situation the processor gets a maximum of workload that takes a maximum time to execute. So, the resulting busy period is the longest possible one. The lowest priority tasks suffer their longest delay in this situation, this is why it is said the worst case scenario. In the same scenario, **PFP\textsubscript{ALAP}** undergoes the same workload and behaves the same way as **PFP\textsubscript{ASAP}** but further delays jobs as long as slack-time is available. So without energy constraints, the worst case scenario of this algorithm is the same as **PFP\textsubscript{ASAP}**.

B. **PFP\textsubscript{ALAP} Study period**

Without energy constraints, the study period of a task set with constrained or implicit deadlines is the processor busy period in the worst case scenario. For **PFP\textsubscript{ASAP}** scheduling, this period is the response time of the first job of the lowest priority task [7]. Following the same logic, the study period of **PFP\textsubscript{ALAP}** algorithm is defined as the period when there is no interference between two successive tasks. Secondly, the next activation period is at least equal or better than a synchronous activation and energy level is $E \geq E_{\text{min}}$. This seems reasonable but it is wrong as illustrated in Figure 2. In this counter-example we can see that despite the system described in Table 4 succeeds during the lower priority task’s response time (time 32), it fails one instance later when the energy level is insufficient to satisfy the energy demand at time 70. To be sure that we will encounter all possible scenarios we must study a longer period bounded by the hyper period. This gives us the possibility to study all possible scenarios. So, the problem of finding a shorter study period is still open.

C. **Optimality ?**

The aim of our work is to build an energy-aware schedulability test for **PFP\textsubscript{ALAP}** that can decide if a given task set is timely and energetically feasible or not, i.e to check if all tasks respect their deadlines while the battery level remains between $E_{\text{max}}$ and $E_{\text{min}}$. We begin with setting an assumption for simplicity: we consider initially the battery to be infinite and not bounded, i.e $E_{\text{max}} = \infty$, then we relax this hypothesis by fixing a bound for the smallest battery capacity necessary to keep the system feasible. A schedulability test is the study of tasks execution according to a scheduling policy during a study period that starts at the critical instant in the worst case energy scenario. In this period we have to test if there is no deadline miss nor energy constraints violation. In our case, the algorithm is **PFP\textsubscript{ALAP}**, the study period is the hyper period and the worst case scenario is the simultaneous activation at the lowest possible battery level i.e the initial energy level $E_0 = E_{\text{min}}$ as previously mentioned. Now it remains to check if all jobs meet their deadlines during the study period, and if the energy recharged during an idle-period is sufficient to meet the energy demand of the busy-period that follows. If the execution satisfies these conditions along the study period, the system is feasible because the periods that follow are at least better or equal to the starting one and the system can execute continuously without constraints violation. To make such verification, we must first calculate the start dates and the length of idle and busy periods and the energy demand of each one.

When the battery reaches its maximum level $E_{\text{max}}$ before the end of an idle period, we cannot continue to charge and
we lose the energy that could be recharged between that moment and the next execution begin. This leads to wast energy if nothing is done to avoid that. In our case this cannot happen because we suppose that $E_{max}$ is infinite. Since $PF_{ALAP}$ postpones jobs executions to charge battery as much as possible before executing, our intuition is to say that such scheduling policy is optimal. We know that all schedulable task sets with $PF_{ASAP}$ are schedulable with $PF_{ALAP}$ so no timing problems can occur and we know also that in $PF_{ALAP}$ idle-periods are as longer as possible which seems better for energy level. These simple intuitions may appear convincing but when we tried to build a formal proof we found a counter-example which invalidates the optimality of $PF_{ALAP}$. Figures 3 and 2 illustrate a counter-example where the task set described in Table I is not schedulable with $PF_{ALAP}$ while it exists a feasible schedule. Indeed, we simulated this system with YARTISS [8]; and this simulation shows that it is feasible until the hyper period. We can see that $PF_{ALAP}$ postpones executions as late as possible which leads in this example to a very long busy period which has a big energy demand and the energy charged during the previous idle period is not sufficient, thus the system failed energetically. When we schedule the same task set with $PF_{ASAP}$ by introducing charging periods as much as necessary when the battery is empty, jobs are scheduled as soon as possible then we avoid some very long busy periods and all jobs meet their deadline and idle periods are always sufficient.

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$C_t$</th>
<th>$E_t$</th>
<th>$D_t$</th>
<th>$D_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>4</td>
<td>216</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>48</td>
<td>48</td>
<td>32</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>16</td>
<td>48</td>
<td>22</td>
</tr>
<tr>
<td>$t_4$</td>
<td>3</td>
<td>186</td>
<td>40</td>
<td>32</td>
</tr>
</tbody>
</table>

Table I

CONTER-EXAMPLE : with $E_{max} = 300$, $E_{min} = 0$ and $P_r(t) = 15$

IV. BATTERY CAPACITY

The design of a system with $E_{max} = \infty$ assumption is not realistic and schedulability test based on such assumption is not very useful, so finding the lowest $E_{max}$ value is necessary. The exact capacity of the battery is difficult to compute because it depends on tasks characteristics and we cannot predict the lowest necessary capacity to satisfy energy demand continuously before executing tasks. So to prevent this, we can solve the problem partially by bounding $E_{max}$ value.

First, knowing that tasks have implicit or constrained deadlines we can limit the study period to the hyper period. Second, the CPU utilization $U$ gives us the proportion of time where CPU is busy, the remaining time the processor is idle and the system is only charging energy. So the highest level of energy that can be reached during a cycle starting from the worst case
scenario of activation and energy is

\[ \text{Bound}_{\text{max}} = \int_0^{(1-U) \times \text{HyperPeriod}} P_r(t) \, dt. \]

This is because we suppose for all tasks that it consumes more energy than we can charge during its execution time, i.e. \( E_i > \int_0^{C_i} P_r(t) \, dt \). This bound is very pessimistic but gives us a sufficient condition. If a task set is schedulable with \( PF_{\text{ALAP}} \) with \( E_{\text{max}} = \infty \) and the given \( E_{\text{max}} \) is greater than or equal to \( \text{Bound}_{\text{max}} \), then it is feasible. By combining the previous test with a battery capacity bound, we have a sufficient condition for the feasibility of a task set.

V. CONCLUSION AND FUTURE WORKS

In this paper we presented some intuitive ideas to solve the problem of the energy harvesting scheduling. Then we showed that simple solutions are not always the best choices despite their obviousness. We invalidated the optimality of \( PF_{\text{ALAP}} \) and showed that the classical study period is no more valid. We also presented a sufficient schedulability test based on \( PF_{\text{ALAP}} \) algorithm and proposed a bound for the lowest necessary battery capacity. It is not an optimal algorithm, so a such schedulability test may be very pessimistic. For this reason we will continue our research on this way by exploring other scheduling policies and heuristics and try to provide more optimistic feasibility conditions. We will study carefully \( PF_{\text{ASAP}} \) and other clairvoyant algorithms and try to establish more optimistic or optimal feasibility conditions.

REFERENCES