Extending Rate Monotonic Analysis with Exact Cost of Preemptions for Hard Real-Time Systems

Patrick Meumeu Yomsi, Yves Sorel

Extending Rate Monotonic Analysis with Exact Cost of Preemptions for Hard Real-Time Systems

Patrick MEUMEU YOMSI
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: patrick.meumeu@inria.fr

Yves SOREL
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: yves.sorel@inria.fr

Abstract

In this paper we study hard real-time systems composed of independent periodic preemptive tasks where we assume that tasks are scheduled by using Liu & Layland's pioneering model following the Rate Monotonic Analysis (RMA). For such systems, the designer must guarantee that all the deadlines of all the tasks are met, otherwise dramatic consequences occur. Certainly, guaranteeing deadlines is not always achievable because the preemption is approximated when using this analysis, and this approximation may lead to a wrong real-time execution whereas the schedulability analysis concluded that the system was schedulable. To cope with this problem the designer usually allows margins which are difficult to assess, and thus in any case lead to a waste of resources. This paper makes multiple contributions. First, we show that, when considering the cost of the preemption during the analysis, the critical instant does **not** *occur upon simultaneous release of all tasks. Second, we provide a technique which counts the exact number of preemptions of each instance for all the tasks of a given system. Finally, we present an RMA extension which takes into account the exact cost due to preemption in the schedulability analysis rather than an approximation, thus yielding a new and stronger schedulability condition which eliminates the waste of resources since margins are not necessary.*

1 Introduction

This paper deals with the problem of executing hard real-time systems found in the domains of automobiles, air traffic control, process control, telecommunications, etc, on a single processor. Such systems often consist of independent periodic preemptive tasks that must meet their deadlines in order to avoid the occurrence of dramatic consequences [1, 2]. Certainly, guaranteeing deadlines cannot always be achieved because the scheduling of the tasks is based on the assumption that the cost of the preemption is approximated within the worst case execution time (WCET) of tasks [3, 4, 5]. In fact, this approximation may be wrong because it is difficult to count the exact number of preemptions of each instance for a given task even though the cost α of one preemption is easy to know for a given processor. Actually, this cost α represents the context switching time that the processor needs when a preemption occurs. The context switch includes the storage of the context as well as the restoration of the context. Since we are interested in embedded systems we only consider predictable processors without a cache or complex internal architecture (e.g. ARM2, etc.) [6, 7]. Therefore, this approximation may lead to a wrong real-time execution whereas the schedulability analysis concluded that the system was schedulable. To cope with this problem the designer usually allows margins which are difficult to assess, and which in any case lead to a waste of resources since the worst case response time is larger than the WCET when an instance of a task has been preempted [8, 9]. Note that the worst-case response time of a task is the longest time it takes, among all instances of that task, to execute each instance from its release time [10]. There have been very few studies addressing this issue of counting the exact number of preemptions. Among them, the most relevant ones are the following. A. Burns, K. Tindell and A. Wellings in [11] presented an analysis that enables the global cost due to preemptions to be factored into the standard equations for calculating the worst case response time of any task, but they achieved that by considering the maximum number of preemptions instead of the exact number. Juan Echagüe, I. Ripoll and A. Crespo also tried to solve the problem of the exact number of preemptions in [12] by constructing the schedule using idle times and counting the number of preemptions. But, they did not really determine the execution overhead incurred by the system due to these preemptions. Indeed, they did not take into account the cost of each preemption during the analysis. Hence, this amounts to considering only the minimum number of preemptions because some preemptions are not considered: those due to the increase in the execution time of the task because of the cost of preemptions themselves.

In this paper, we first show that the critical instant [3] does not occur when all tasks are released simultaneously if we consider the cost of the preemption during the analysis. Second, we propose a new scheduling algorithm which counts the exact number of preemptions of each instance for all tasks. Finally, we propose a new and stronger schedulability condition than Liu & Layland's condition, which takes into account the exact cost due to preemption in the schedulability analysis. This new condition always guarantees a correct execution and eliminates any waste of resources since no margins are necessary.

We assume that tasks are scheduled by using Liu & Layland's pioneering model according to Rate Monotonic Analysis (RMA) [3, 13]. That is to say, we are in the fixed priority context and the highest fixed priority is assigned to the task with the shortest period [14, 15]. When two tasks have the same period they are scheduled arbitrarily [16]. We consider a set of n independent periodic preemptive tasks $\tau_i, 1 \leq i \leq n$. Each task τ_i is an infinite sequence of instances [1] τ_i^k, $k \in \mathbb{N}^+$, and is characterized by a WCET C_i, not including the approximation of the cost of the preemption, a period T_i, and a release time relative to 0, r_i. This means that instances corresponding to task τ_i are released at times $r_i + kT_i$, $k \geq 0$. The instance released at time $r_i + kT_i$ has $r_i + (k+1)T_i$ as its deadline, i.e. the release time of the next instance. We re-index tasks in such a way that $T_1 \leq T_2 \leq \cdots \leq T_n$. Consequently, τ_i receives priority i [2] and we assume that tasks are ready to run at their release times (idle time is forbidden in the presence of ready tasks).

For the sake of readability and without any loss of generality, from now on, although it is not realistic, we consider the cost of one preemption for the processor to be $\alpha = 1$ time unit in all the examples. This high cost of preemptions in terms of the execution time of tasks is used to illustrate the impact of not accounting for preemptions correctly.

In addition, it is worth noticing that the analysis performed here would work even if the preemption cost is not a constant.

The remainder of this paper is structured as follows: section 2 gives a counterexample on the critical instant when the cost of preemption is considered. Section 3 describes the model and gives the notations used throughout the paper. Section 4 provides the definitions we need to take into account the exact cost of preemption in the schedulability analysis presented in section 5. That section explains in detail, on the one hand, our scheduling algorithm which counts the exact number of preemptions and, on the other hand, derives the new schedulability condition. The complexity of our algorithm is discussed in section 6. We conclude and propose future work in section 7.

[1] Throughout the paper all subscripts refer to tasks whereas all superscripts refer to instances.

[2] 1 represents the highest priority.

2 Critical instant

The critical instant when the cost of the preemption is approximated within the WCET of tasks is such that the release time of the first instance of tasks occurs simultaneously [3], that is to say $r_i = 0$ for all $1 \leq i \leq n$. However, this is not necessarily the critical instant when the cost of preemptions is considered, see the counterexample depicted in figure 1 (the "■" represents the preemption cost).

Tasks	r_i	C_i	T_i
τ_1	0	2	5
τ_2	0	2	8

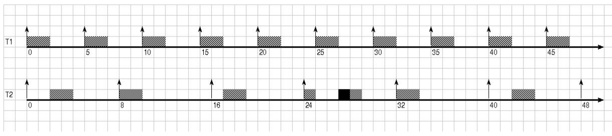

Figure 1. Schedule under RMA with the cost of preemption considered, $r_i = 0$

In figure 1 the response time (4 time units) of task τ_2 in its first instance (corresponding to the critical instant) is shorter than the response time (5 time units) of its fourth instance. This is because τ_2 has been preempted in the fourth instance and then, the cost of the preemption has been added to the WCET without any approximation, and used to compute the response time in that instance.

If the first instances of all tasks are released simultaneously, then this is repeated every hyperperiod H, thus as stated in [3, 1] it is sufficient to perform the schedulability analysis in the interval $[0, H]$. H is the least common multiple of the periods of the tasks: $H = lcm\{T_1, T_2, \cdots, T_n\}$. For this reason, in this paper, we assume that all tasks are released simultaneously. Since the worst case response time of a task may not occur in the first instance (*see* figure 1), we consider all instances of a task within a hyperperiod, and perform the schedulability analysis only within the first hyperperiod.

Because we intend to take into account the exact cost of the preemption, and because all tasks, except the first one, may be preempted, the proposed technique gives a schedulability condition for each task individually according to tasks with higher priority. Our scheduling algorithm calculates the exact number of preemptions per instance of every task. This individual analysis leads, at the end, to a schedulability condition for all the tasks.

3 Model and Notations

Throughout this paper, all timing characteristics in our model are assumed to be non-negative integers, i.e. they are multiples of some elementary time interval (for example the "CPU tick", the smallest indivisible CPU time unit).

Since all tasks except the one with the highest priority may be preempted, the execution time of a task may vary from one instance to another. We call *preempted execution time* (PET) the WCET augmented with the exact cost due to preemptions for each instance of a task within a hyperperiod. Thus, the PET denoted C_i^k for instance τ_i^k of task τ_i is greater than or equal to its WCET C_i. It depends on the instance and on the number of preemptions occuring in that instance. Its calculation will be detailed below.

The following model (depicted in figure 2) is an extension, with the exact cost of preemption, of the classical model [3] for systems of tasks executed on a single processor.

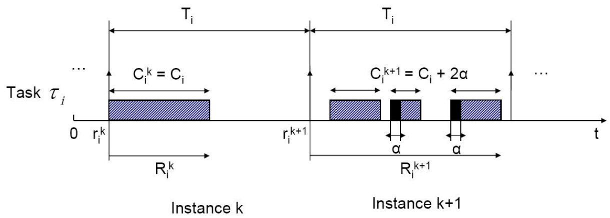

Figure 2. Model

$\tau_i = (C_i, T_i)$: A task
T_i: Period of τ_i
C_i: WCET of τ_i not including the preemption approximation, $C_i \leq T_i$
α: Temporal cost of one preemption for a given processor
τ_i^k: The k^{th} instance of τ_i
$N_p(\tau_i^k)$: Exact number of preemptions of τ_i in τ_i^k
$C_i^k = C_i + N_p(\tau_i^k) \cdot \alpha$: PET of τ_i with its preemption cost in τ_i^k
$r_i^k = (k-1)T_i$: Release time of τ_i^k
R_i^k: Response time of τ_i^k
R_i: Worst-case response time of τ_i

From the point of view of task τ_i, since it may only be preempted by higher priority tasks, we define the *hyperperiod at level i*, H_i, which is given by $H_i = lcm\{T_j\}_{\tau_j \in hep(\tau_i)}$, where $hep(\tau_i)$ is the set of tasks with a priority higher than or equal to task τ_i. Hence, task τ_i is released σ_i times in each hyperperiod at level i starting from 0, with

$$\sigma_i = \frac{H_i}{T_i} = \frac{lcm\{T_j\}_{\tau_j \in hep(\tau_i)}}{T_i} \tag{1}$$

The total utilization factor is usually given by

$$U_n = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2}$$

Recall that in (2) C_i does not include the approximation of the cost of the preemption for task τ_i. If $U_n > 1$ then the task set is not schedulable with any algorithm [17]. Thus, a set of n tasks may be *schedulable* if and only if $U_n \leq 1$ [18, 19]. Indeed, U_n can be lower than or equal to 1 and the system not schedulable.

According to the number of preemptions $N_p(\tau_i^k)$ of task $\tau_i = (C_i, T_i)$ in each instance τ_i^k, its PET C_i^k may be different from one instance to another, except for the task with the highest priority τ_1 which can never be preempted. However, because task τ_i may only be preempted by the set of tasks with a priority higher than τ_i denoted $hp(\tau_i)$ [3], then there are exactly σ_i different PETs for task τ_i. In other words, from the point of view of any task τ_i, $1 \leq i \leq n$, there exists a function $\pi : \mathbb{N}^+ \times \mathbb{N}^+ \longrightarrow \mathbb{N}^{+\sigma_i} \times \mathbb{N}^+$, defined as $\pi(C_i, T_i) = \pi(\tau_i) = ((C_i^1, C_i^2, \cdots, C_i^{\sigma_i}), T_i)$, which maps the WCET C_i of task τ_i into its respective PET C_i^k in each instance τ_i^k. Therefore, each task $\tau_i = (C_i, T_i)$ has an image $\tau_i' = \left((C_i^1, C_i^2, \cdots, C_i^{\sigma_i}), T_i\right)$. Consequently, we define the *exact total utilization factor* to be

$$U_n^* = \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_n + \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \tag{3}$$

Remark that if $\alpha = 0$, then equation (3) reduces to the classical total utilization factor U_n when the global cost due to preemption is approximated within the WCET of tasks. Therefore, the global cost due to preemptions incurred by the system is

$$\varepsilon_n = \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \tag{4}$$

Now we have to calculate $N_p(\tau_i^k)$ for all $k = 1, \cdots, \sigma_i$ and for all $i = 1, \cdots, n$. To do so, let us recall some useful algebra that we need to achieve this goal.

4 Definitions

For a given set of n tasks, we define the *exact total utilization factor at level j*, $1 \leq j \leq n$ to be

$$U_j^* = \sum_{i=1}^{j} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_j + \sum_{i=1}^{j} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \tag{5}$$

It is worth noticing that since we are in a fixed priority context, and thus we carry out the schedule from the highest priority task towards lower priority tasks, then to every instance τ_i^k of a task $\tau_i = (C_i, T_i)$ is associated an ordered set of T_i time units where some are already executed because of the execution of a higher priority task, and the others are still available for the execution of task τ_i in that instance. We call this ordered set which describes the state of each instance τ_i^k a *T_i-mesoid*. We denote a time unit already executed by an "*e*" and a time unit still available by an "*a*". Obviously, the switch from an *a* to an *e* represents a preemption if the WCET of the task under consideration is strictly greater than the cardinal of the

[3] $hp(\tau_i)$ is the set of tasks with a priority higher than task τ_i.

sub-set corresponding to the first sequence of a. According to the remaining execution time this situation may occur again. For example, $\{e,e,e,a,a,a,e,e,a,a,e,a,a\}$ is a mesoid where the first 3 time units have already been executed, the next 3 time units are available, followed again by 2 already executed, then 2 available followed by one already executed and which ends with 2 available. For the sake of clarity and without any loss of generality, we call a sub-set corresponding to a sequence of consecutive time units already executed a *consumption*, and we represent it by its cardinal inside brackets (c), with $c \in \mathbb{N}^+$. In addition, we enumerate the sequence of available time units according to the natural numbers. This enumeration is done from the end of the first sequence of time units already executed in that instance. Each of these natural numbers corresponds to the number of available time units since the end of the first consumption. They represent all the possible PETs of the task under consideration in the corresponding instance. Each of these natural numbers is called an *availability*. Thus, the previous 13-mesoid can be re-writen as: $\{(3),1,2,3,(2),4,5,(1),6,7\}$. It has three consumptions 3, 2, 1, and seven availabilities 1, 2, 3, 4, 5, 6, 7. If the PET of the task under consideration is equal to 6 then there are two preemptions. Notice that the sum of all the consumptions of a mesoid and the highest availability in that mesoid, is equal to the period of the task under consideration. From the point of view of task $\tau_i = (C_i, T_i)$, there are as many T_i-mesoids as instances in the hyperperiod H_i at level i, because task τ_i may only be preempted by tasks in $hp(\tau_i)$. Therefore, there are σ_i T_i-mesoids in H_i which will form a sequence of T_i-mesoids. We call $\mathcal{L}_i^b = \left\{\mathcal{M}_i^{b,1}, \mathcal{M}_i^{b,2}, \cdots, \mathcal{M}_i^{b,\sigma_i}\right\}$ the sequence of σ_i T_i-*mesoids* **before** τ_i is scheduled. For example, $\mathcal{L}_i^b = \{\{(5),1,2,3,(2),4\},\{1,2,(3),3,4,(3),5\}\}$ is a sequence of $\sigma_i = 2$ 11-mesoids. The process for building the sequence $\mathcal{L}_i^b$ of task τ_i will be detailed later on in this paper.

Still, from the point of view of task τ_i, we define for each mesoid $\mathcal{M}_i^{b,k}, 1 \le k \le \sigma_i$ of the sequence $\mathcal{L}_i^b$ the corresponding *universe* X_i^k of τ_i to be the set which consists of all the availabilities of $\mathcal{M}_i^{b,k}$. That is to say, all the possible values that C_i^k can take in $\mathcal{M}_i^{b,k}$. Recall that C_i^k denotes the PET of τ_i in τ_i^k, the k^{th} instance of τ_i. For the previous example of a sequence of 11-*mesoids*, $\mathcal{M}_i^{b,2} = \{1,2,(3),3,4,(3),5\}$, and thus we have $X_i^2 = \{1,2,3,4,5\}$.

Task τ_i will be said to be *potentially schedulable* if and only if

$$\begin{cases} U_{i-1}^* + \dfrac{C_i}{T_i} \le 1 \\ \\ \mathrm{C}_i \in X_i^k \quad \forall k \in \{1,\cdots,\sigma_i\} \end{cases} \tag{6}$$

The first equation of (6) verifies that the minimum exact total utilization factor at level i is less than or equal to 1. Indeed, $U_{i-1}^* + \dfrac{C_i}{T_i} \le U_i^*$ because all WCET $C_i \le C_i^k, \forall k \ge 1$, and when task τ_i is shedulable $U_i^* \le 1$ must hold. The σ_i other equations verify that C_i belongs to all the universes.

Since $C_i \in \{1,2,\cdots,T_i\}, \forall 1 \le i \le n$, let us define the following binary relation on each instance

$\mathcal{R}$: "WCET C_{γ_1} leads to the same number of preemptions as WCET C_{γ_2}", $C_{\gamma_1}, C_{\gamma_2} \in \{1,2,\cdots,T_i\}$

$\mathcal{R}$ is clearly an equivalence relation on $\{1,2,\cdots,T_i\}$ (reflexive, symmetric, transitive). Now, since $X_i^k \subseteq \{1,2,\cdots,T_i\}, \forall 1 \le k \le \sigma_i$, thus $\mathcal{R}$ is also an equivalence relation on X_i^k, $\forall 1 \le k \le \sigma_i$ and each $X_i^k, k = 1,\cdots,\sigma_i$ together with $\mathcal{R}$ is a *setoid* [4]. From now on, we consider only the restriction of $\mathcal{R}$ on $X_i^k, k = 1,\cdots,\sigma_i$ because X_i^k represents all the available time units in instance τ_i^k.

The *equivalence classes* of each universe are the subsets of availabilities determined by two consecutive consumptions in the associated mesoid. In the remainder of this paper, we call these equivalence classes the *cells* of the universe. Hence, for the above example, we have

X_i^1: $[0]^1 = \{1,2,3\}$ and $[1]^1 = \{4\}$

X_i^2: $[0]^2 = \{1,2\}$ and $[1]^2 = \{3,4\}$ and $[2]^2 = \{5\}$

where for $m \in \mathbb{N}$ and $1 \le k \le \sigma_i$, $[m]^k$ denotes the subset of X_i^k composed of the availabilities which are preempted m times. Thus, for the previous example, $\mathcal{L}_i^b$ can also be written as

$$\mathcal{L}_i^b = \{\{(5),\overbrace{1,2,3}^{[0]^1},(2),\overbrace{4}^{[1]^1}\},\{\overbrace{1,2}^{[0]^2},(3),\overbrace{3,4}^{[1]^2},(3),\overbrace{5}^{[2]^2}\}\}$$

This means for task τ_i that its PET $C_i^k \in X_i^k$, i.e. in its k^{th} instance, $k = 1,2$, should not exceed 4 in the first instance, and 5 in the second instance otherwise task τ_i cannot be schedulable. We call $\mathcal{L}_i^a = \left\{\mathcal{M}_i^{a,1}, \mathcal{M}_i^{a,2}, \cdots, \mathcal{M}_i^{a,\sigma_i}\right\}$ the sequence of σ_i T_i-*mesoids* of task τ_i **after** τ_i is scheduled. $\mathcal{L}_i^a$ is a function of $\mathcal{L}_i^b$ which itself is a function of $\mathcal{L}_{i-1}^a$, both detailed as follows.

We build the sequence $\mathcal{L}_i^b$ for task τ_i by using an index ζ which enumerate, according to natural numbers, the time units in the sequence $\mathcal{L}_{i-1}^a$ of task τ_{i-1} **after** τ_{i-1} is scheduled. This enumeration is done whether the time units have already been consumed or are still available. ζ starts from the first time unit of the first mesoid $\mathcal{M}_{i-1}^{a,1}$ towards the last time unit of the last mesoid $\mathcal{M}_{i-1}^{a,\sigma_{i-1}}$, and then circles around to the beginning of the first mesoid $\mathcal{M}_{i-1}^{a,1}$ again. This process of counting is thus **cyclic**. Each time $\zeta = T_i$, a T_i-mesoid is obtained for the sequence $\mathcal{L}_i^b$ and then the next T_i-mesoid is obtained by starting to count again from the next time unit to the current one. This process is repeated until we get the σ_i T_i-mesoids of $\mathcal{L}_i^b$. Since task τ_1 is the highest priority task, $hep(\tau_1) = \{\tau_1\}$ and thus $\sigma_1 = \dfrac{H_1}{T_1} = 1$

[4] A setoid is a set equipped with an equivalence relation.

thanks to equation (1). Moreover, because it is never preempted, we have $\mathcal{L}_1^b = \left\{\mathcal{M}_1^{b,1}\right\} = \{\{1,2,\cdots,T_1\}\}$ and $\mathcal{L}_1^a = \left\{\mathcal{M}_1^{a,1}\right\} = \{\{(C_1),1,2,\cdots,T_1 - C_1\}\}$.

The sequence $\mathcal{L}_i^a$ is deduced from the sequence $\mathcal{L}_i^b$ because all the available time units will have been consumed up to the response time (detailed later on) within each mesoid $\mathcal{M}_i^{b,k}, k = 1,\cdots,\sigma_i$ of task τ_i **after** τ_i is scheduled. Notice that the response time in each mesoid depends on π for task τ_i.

To summarize, for every task τ_i, we have

$$\tau_i : \begin{cases} \mathcal{L}_i^b = \left\{\mathcal{M}_i^{b,1}, \mathcal{M}_i^{b,2}, \cdots, \mathcal{M}_i^{b,\sigma_i}\right\} \\ \\ \mathcal{L}_i^a = \left\{\mathcal{M}_i^{a,1}, \mathcal{M}_i^{a,2}, \cdots, \mathcal{M}_i^{a,\sigma_i}\right\} \end{cases}$$

Both $\mathcal{L}_i^b$ and $\mathcal{L}_i^a$ consist of a finite number σ_i of T_i-mesoids in each sequence.

5 The proposed approach

In this section we outline our approach that leads to a new and stronger schedulability condition than the condition proposed by Liu & Layland [3], Joseph and Pandya [20], Lehoczky et al. [5], Audsley et al.[21], etc. in the sense that it takes the cost of preemption accurately into account in the schedulability analysis rather than using an approximation. The intuitive idea behind our approach uses a system of arithmetic for integers, where numbers "wrap around" after they have reached a certain value: the period of the task under consideration. In other words, our approach uses a modulo T arithmetic where T is the period of a task.

5.1 Scheduling of two tasks

Let us motivate the general result of our approach by considering the simple case of the scheduling problem of two tasks $\tau_1 = (C_1, T_1)$ and $\tau_2 = (C_2, T_2)$, with $T_1 \leq T_2$. Under RMA, τ_1 is assigned the higher priority. This latter statement implies that **before** τ_1 is scheduled, its WCET C_1 can potentially take any value from 1 up to the value of its period T_1, therefore $\mathcal{L}_1^b = \left\{\mathcal{M}_1^{b,1}\right\} = \{\{1,2,\cdots,T_1\}\}$. Since task τ_1 is never preempted, thus $C_1^k = C_1$, $\forall k \geq 1$ and $\sigma_1 = 1$ and $\tau_1^{'} = \pi(\tau_1) = ((C_1), T_1)$. In addition, its response time is also equal to C_1. Hence, **after** τ_1 is scheduled, it has consumed C_1 time units, and thus there remain $T_1 - C_1$ availabilities in each of its instances. Consequently, the corresponding T_1-*mesoids* associated to task τ_1 are given by

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \left\{\mathcal{M}_1^{b,1}\right\} = \{\{1,2,\cdots,T_1\}\} \\ \\ \mathcal{L}_1^a = \left\{\mathcal{M}_1^{a,1}\right\} = \{\{(C_1),1,2,\cdots,T_1 - C_1\}\} \end{cases}$$

Now, the challenge is to schedule task τ_2 by taking into account the exact cost of preemptions. Thanks to everything we have presented up to now, the construction of $\mathcal{L}_2^b$ consists of $\sigma_2 = \dfrac{H_2}{T_2}$ T_2-*mesoids*. Furthermore, the sequence $\mathcal{L}_2^b$ is built by using the index ζ and enumerating **cyclically** the time units in the sequence $\mathcal{L}_1^a$. The construction of $\mathcal{L}_2^b$ is based on the intuitive idea of a modulo T_2 arithmetic. After the construction of $\mathcal{L}_2^b$, we can easily determine the corresponding universe X_2^k to each T_2-mesoid $\mathcal{M}_2^{b,k}$, $k = 1,\cdots,\sigma_2$. Thus, thanks to equation (6), task τ_2 is *potentially schedulable* if and only if

$$\begin{cases} U_1^* + \dfrac{C_2}{T_2} \leq 1 \\ \\ C_2 \in X_2^k \quad \forall k \in \{1,\cdots,\sigma_2\} \end{cases} \tag{7}$$

We give the following example in order to illustrate these conditions. Let us consider a set of two tasks τ_1 and τ_2 with $T_1 = 6$, $T_2 = 8$, and $C_1 = 2$, $C_2 = 3$. We have

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1,2,3,4,5,6\}\} \\ \\ \mathcal{L}_1^a = \{\mathcal{M}_1^{a,1}\} = \{\{(2),1,2,3,4\}\} \end{cases}$$

Since $\sigma_2 = \dfrac{H_2}{T_2} = 3$, thus we derive $\mathcal{L}_2^b$ which consists of a sequence of three 8-mesoids by using the index ζ as explained in the previous section on the sequence $\mathcal{L}_1^a$. We obtain

$$\begin{aligned} \mathcal{L}_2^b &= \left\{\mathcal{M}_2^{b,1}, \mathcal{M}_2^{b,2}, \mathcal{M}_2^{b,3}\right\} \\ &= \{\{(2),1,2,3,4,(2)\},\{1,2,3,4,(2),5,6\}, \\ &\quad \{1,2,(2),3,4,5,6\}\} \end{aligned}$$

For each 8-*mesoid* $\mathcal{M}_2^{b,k}, 1 \leq k \leq 3$, composing $\mathcal{L}_2^b$, we build the corresponding universe $X_2^k, 1 \leq k \leq 3$. These universes are given by

$$\tau_2 : \left\| \begin{array}{l} \mathrm{X}_2^1 = \{1,2,3,4\} \\ \mathrm{X}_2^2 = \{1,2,3,4,5,6\} \\ \mathrm{X}_2^3 = \{1,2,3,4,5,6\} \end{array} \right.$$

From these universes, we deduce that task τ_2 is potentially schedulable because for each resulting universe X_2^k, we have

$$\begin{cases} U_1^* + \dfrac{C_2}{T_2} = \dfrac{2}{6} + \dfrac{3}{8} \leq 1 \\ \\ 3 \in X_2^k \quad \forall k \in \{1,\cdots,\sigma_2\} \end{cases}$$

Now, thanks to the equivalence relation $\mathcal{R}$ on each X_2^k for $k = 1,\cdots,3$, the *cells* of each universe are given by

for universe X_2^1: $[0]^1 = \{1,2,3,4\}$

for universe X_2^2: $[0]^2 = \{1,2,3,4\}$ and $[1]^2 = \{5,6\}$

for universe X_2^3: $[0]^3 = \{1,2\}$ and $[1]^3 = \{3,4,5,6\}$

where for $m \in \mathbb{N}$ and $1 \leq k \leq \sigma_2$, $[m]^k$ denotes the subset of X_2^k composed of the availabilities which are preempted m times. Thus, for this example, $\mathcal{L}_2^b$ can also be written as

$$\mathcal{L}_2^b = \{\{(2),\overbrace{1,2,3,4}^{[0]^1},(2)\},\{\overbrace{1,2,3,4}^{[0]^2},(2),\overbrace{5,6}^{[1]^2}\}, \{\overbrace{1,2}^{[0]^3},(2),\overbrace{3,4,5,6}^{[1]^3}\}\}$$

Here we have all we need to calculate the exact number of preemptions $N_p(\tau_2^k)$ and then the corresponding PET C_2^k of task τ_2 in its k^{th} instance, $1 \leq k \leq \sigma_2$.

Since task τ_2 is potentially schedulable (equation (7) holds), thus, its WCET C_2 belongs to one and only one cell $[\theta_1]^k$ in each universe $X_2^k, k = 1,\cdots,\sigma_2$ (see figure 4 when $i = 2$) since each $(X_2^k, \mathcal{R})$ is a setoid. As such, the PET C_2^k is different from the actual value of the WCET C_2 in the associated mesoid as soon as task τ_2 must be preempted at least once. This occurs when $C_2 \in X_2^k \backslash [0]^k$ for any k with $1 \leq k \leq \sigma_2$, (see figure 5 when $i = 2$).

In each universe $X_2^k, 1 \leq k \leq \sigma_2$, the number of preemptions $N_p(\tau_2^k)$ and the PET C_2^k of task τ_2 are computed by using the following algorithm.

Initialization:

$$\begin{cases} C_2^{k,1} = C_2 \\ B^{k,1} = C_2 \\ q^{k,1} = \theta_1 \\ A^{k,1} = \sum_{m=0}^{\theta_1 - 1} card([m]^k) \\ r_{k,1} = C_2^{k,1} - A^{k,1} \end{cases}$$

For $l \geq 1$, we compute

$$B^{k,l+1} = \sum_{j=1}^{l} A^{k,j} + (r_{k,l} + \theta_l \cdot \alpha) \tag{8}$$

By using the same idea as for a fixed-point algorithm, this computation stops as soon as either two consecutive values of $B^{k,j}, j \geq 1$, belong to the same cell or there exists $\mu_1 \geq 1$ such that $B^{k,\mu_1} > card(X_2^k)$. Figure 6 when $i = 2$ illustrates the same idea as for a fixed point algorithm. In this latter case, task τ_2 is not schedulable because the period (deadline) of the task is thus exceeded. Actually, if $B^{k,l+1} \leq card(X_2^k)$, then $\exists \theta_{l+1} \geq 0$ such that

$$B^{k,l+1} \in [\theta_1 + \cdots + \theta_{l+1}]^k$$

If $\theta_{l+1} = 0$ then $B^{k,l+1}$ and $B^{k,l}$ belong to the same cell, therefore expression (9) holds with $\mu_2 = l+1$ and $N_p(\tau_2^k)$ is given by (10), else if $\theta_{l+1} \neq 0$, then we compute

$$\begin{cases} C_2^{k,l+1} = r_{k,l} + \theta_l \cdot \alpha \\ q^{k,l+1} = \theta_{l+1} \\ A^{k,l+1} = \sum_{m=\theta_1+\cdots+\theta_l}^{\theta_1+\cdots+\theta_{l+1}-1} card([m]^k) \\ r_{k,l+1} = C^{k,l+1} - A^{k,l+1} \end{cases}$$

and thus we derive the next value of $B^{k,j}$. The algorithm is stopped as soon as

$$\exists \mu_2 \geq 1 \quad \text{such that} \quad \theta_{\mu_2} = 0 \tag{9}$$

and therefore

$$N_p(\tau_2^k) = \sum_{j=1}^{\mu_2 - 1} q^{k,j} \tag{10}$$

Thanks to equation (10), for each $k = 1,\cdots,\sigma_2$, we compute the PET C_2^k of task τ_2 in X_2^k, i.e. in its k^{th} instance, including its exact preemption cost. Figure 6 when $i = 2$, in addition to illustrate the same idea as for a fixed point algorithm, also shows the PET of the task τ_i in instance τ_i^k.

$$C_2^k = C_2 + N_p(\tau_2^k) \cdot \alpha \tag{11}$$

Consequently, the image of τ_2 by function π is given by

$$\tau_2^{'} = \pi(\tau_2) = \left((C_2^1, C_2^2, \cdots, C_2^{\sigma_2}), T_i\right) \tag{12}$$

The response time $R_2^k, 1 \leq k \leq \sigma_2$ of task τ_2 in its k^{th} instance, i.e. in the k^{th} T_2-*mesoid* is obtained by summing C_2^k with all the consumptions appearing before C_2^k in the corresponding mesoid. Once this has been done, the worst-case response time R_2 of task τ_2 is given by

$$R_2 = max_{\{1 \leq k \leq \sigma_2\}}(R_2^k)$$

The sequence $\mathcal{L}_2^a$ is deduced from sequence $\mathcal{L}_2^b$ by updating the latter since all time units up to the response time have now been consumed in every mesoid. Hence, by using expression (3), the exact total utilization factor of the CPU is given by

$$U_2^* = \sum_{i=1}^{2} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_2 + \frac{1}{\sigma_2} \left(\sum_{k=1}^{\sigma_2} \frac{N_p(\tau_2^k) \cdot \alpha}{T_2} \right) \tag{13}$$

Let us illustrate this result on the previous example. We still assume $\alpha = 1$ to be the cost of one preemption for the processor in order to give a clear indication of the impact of the preemption. We recall that $C_2 = 3$, task τ_2 is potentially schedulable, and

$$\mathcal{L}_2^b = \{\{(2),\overbrace{1,2,3,4}^{[0]^1},(2)\},\{\overbrace{1,2,3,4}^{[0]^2},(2),\overbrace{5,6}^{[1]^2}\}, \{\overbrace{1,2}^{[0]^3},(2),\overbrace{3,4,5,6}^{[1]^3}\}\}$$

In both the first and second universes, $C_2 \in [0]^k, k = 1,2$; thus $C_2^1 = C_2^2 = C_2$ whereas in the third universe, $C_2 \in [1]^3$. The computation of $N_p(\tau_2^3)$ is summarized in the following table.

From the second column of table 1, we get $N_p(\tau_2^3) = 1$ and thus we obtain $C_2^3 = 3 + 1 \cdot 1 = 4$. Hence, the image of task τ_2 by function π is given by $\tau_2^{'} = \pi(\tau_2) = ((3,3,4),8)$. Therefore, $U_2^* = \frac{2}{6} + \frac{1}{3}\left(\frac{3+3+4}{8}\right) = 0.750$ whereas

Table 1. computation of $N_p(\tau_2^3)$

Steps	$q^{3,l}$	$C_2^{3,l}$	$A^{3,l}$	$r_{3,l}$	$B^{3,l}$
1	1	3	2	1	4
2	0	2	2	0	4

$U_2 = \dfrac{2}{6} + \dfrac{3}{8} = 0.708$. The response times of task τ_2 in each mesoid thanks to our previous definition are given by

$$R_2^1 = 3+2 = 5,\ R_2^2 = 3, \text{ and } R_2^3 = 4+2 = 6$$

Hence, from this approach we can obviously deduce the worst-case response time R_2 of task τ_2: $R_2 = 6$. Task τ_2 is schedulable and its response time $R_2^k, 1 \leq k \leq \sigma_2$ in its k^{th} instance, i.e. in the k^{th} T_2-*mesoid*, is the first consumption in $\mathcal{M}_2^{a,k}$ of sequence $\mathcal{L}_2^a$. Figure 3 depicts the schedule of this example taking into account the exact cost of preemtion.

For more than two tasks notice that $\mathcal{L}_2^a$ is deduced from $\mathcal{L}_2^b$ by updating the latter as follows.

$$\begin{aligned}\mathcal{L}_2^b \quad = \quad & \{\{(2), \overbrace{1,2,3,4}^{[0]^1}, (2)\}, \{\overbrace{1,2,3,4}^{[0]^2}, (2), \overbrace{5,6}^{[1]^2}\}, \\ & \{\overbrace{1,2}^{[0]^3}, (2), \overbrace{3,4,5,6}^{[1]^3}\}\}\end{aligned}$$

$$\mathcal{L}_2^a = \{\{(5), 1, (2)\}, \{(3), 1, (2), 2, 3\}, \{(6), 1, 2\}\}$$

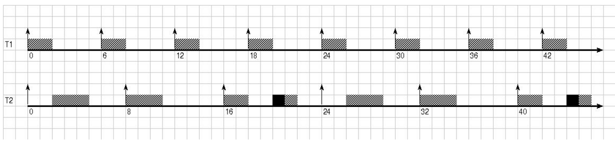

Figure 3. Execution of two tasks following RMA with exact cost of preemption

5.2 Scheduling of $n > 2$ tasks

The strategy that we will adopt in this section to calculate both the exact number of preemptions and the PETs of a given task in each of its instances is the generalization to a system of $n > 2$ tasks of everything we have presented in the previous section for the simple case of two tasks. Indeed, the basic idea behind this approach consists, for each task, in filling availabilities in each mesoid with slices (cardinal of cells) of its PET which takes into account the cost of the exact number of preemptions necessary for its schedule. Recall that at each preemption occurence, α time units add to the remaining execution time of the instance of the task under consideration.

Before going through our proposed algorithm, we recall the exact total utilization factor at level j, U_j^*, with $1 \leq j < n$,

$$U_j^* = \sum_{i=1}^{j} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_j + \sum_{i=1}^{j} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \tag{14}$$

Without any loss of generality, we assume that all tasks have different periods, that is to say $T_i < T_j$ for $1 \leq i < j \leq n$. A sub-system of tasks $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq p}$, with $1 \leq p < n$, is said to be *maximal* when the *the exact total utilization factor at level p* is smaller than or equal to 1, and the *exact total utilization factor at level* $(p+1)^{th}$ is strictly larger than 1, this occurs when

$$U_p^* \leq 1 \quad \text{and} \quad U_{p+1}^* > 1 \tag{15}$$

This means that the sub-system $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq p}$ is the largest sub-system schedulable on the processor according to RMA.

5.3 Scheduling algorithm

We assume that the first $i-1$ tasks with $2 \leq i \leq n$ have already been scheduled, and that we are about to schedule the i^{th} task, i.e. task τ_i, which is potentially schedulable, i.e.

$$\begin{cases} U_{i-1}^* + \dfrac{C_i}{T_i} < 1 \\ \\ C_i \in X_i^k \quad \forall k \in \{1, \cdots, \sigma_i\} \end{cases}$$

As in the previous section for the construction of $\mathcal{L}_2^b$ using index ζ on the sequence $\mathcal{L}_1^a$, the sequence $\mathcal{L}_i^b$ of task τ_i is built thanks to index ζ on the sequence $\mathcal{L}_{i-1}^a$ of task τ_{i-1}. The sequence $\mathcal{L}_i^b$ consists of σ_i T_i-*mesoids* $\mathcal{M}_i^{b,k}$ with $k = 1, \cdots, \sigma_i$ since task τ_i may only be preempted by tasks belonging to $hp(\tau_i)$. Therefore, we can determine the universes $X_i^k \quad \forall k \in \{1, \cdots, \sigma_i\}$ when the sequence $\mathcal{L}_{i-1}^a$ is known. Again, the response time $R_i^k, 1 \leq k \leq \sigma_i$ of task τ_i in its k^{th} instance, i.e. in the k^{th} T_i-*mesoid* will be obtained by summing C_i^k with all consumptions prior to C_i^k in the corresponding mesoid. The worst-case response time R_i of task τ_i will be given by

$$R_i = max_{\{1 \leq k \leq \sigma_i\}}(R_i^k)$$

This equation leads us to say that task τ_i will be schedulable if and only if

$$R_i \leq T_i \tag{16}$$

Again, $\mathcal{L}_i^a$ will be deduced from $\mathcal{L}_i^b$ by updating the latter since all time units up to the response time will have been consumed in each mesoid. For the sake of clarity, note that when updating $\mathcal{L}_i^b$, whenever there are two consecutive consumptions in the same *mesoid*, this amounts to considering only one consumption which is the sum of the previous consumptions. That is to say that after determining the response time of task τ_i in its k^{th} mesoid, if $\mathcal{M}_i^{a,k} = \{(c_1), (c_2), 1, 2, \cdots\}$, then this is equivalent to

$\mathcal{M}_i^{a,k} = \{(c_1 + c_2), 1, 2, \cdots\}$ without any loss of generality.

Below, we present our scheduling algorithm which, for a given task on the one hand, counts the exact number of preemptions in each of its instances, and on the other hand, provides its PET in each of its instances in order to take the cost of the preemption into account accurately in the schedulability condition. It has the twelve following steps. Since the highest priority task, namely task τ_1, is never preempted, the loop starts from the index of the second highest priority task, namely task τ_2 as we carry out the schedule towards lower priority tasks.

1: **for** $i = 2$ to n **do**

2: Compute the number σ_i of times that task $\tau_i = (C_i, T_i)$ is released in the hyperperiod at level i

$$\sigma_i = \frac{H_i}{T_i} = \frac{lcm\{T_j\}_{\tau_j \in hep(\tau_i)}}{T_i}$$

Recall that $H_i = lcm\{T_1, T_2, \cdots, T_i\}$

3: Build the sequence $\mathcal{L}_i^b$ of T_i-*mesoids* of task τ_i before it is scheduled. This construction consists of σ_i T_i-*mesoids* $\mathcal{M}_i^{b,k}$ with $k = 1, \cdots, \sigma_i$, and is based on a modulo T_i arithmetic using the the index ζ on the sequence $\mathcal{L}_{i-1}^a$.

4: For each T_i-*mesoid* $\mathcal{M}_i^{b,k}$ resulting from the previous step, build the corresponding universe X_i^k which is composed of the set of all availabilities in $\mathcal{M}_i^{b,k}$. Notice that this set corresponds to the set of all possible values that the PET C_i^k of task τ_i can take in $\mathcal{M}_i^{b,k}$.

5: Build all the cells for each universe X_i^k. A cell of X_i^k is composed of the subset of availabilities determined by two consecutive consumptions in the associated mesoid $\mathcal{M}_i^{b,k}$.

6: Compute both the exact number of preemptions and the PET C_i^k of task τ_i in each universe $X_i^k, 1 \leq k \leq \sigma_i$, resulting from the previous step thanks to the algorithm inlined in this step. This algorithm is necessary because, since τ_i is potentially schedulable, i.e. its WCET C_i belongs to one and only one cell $[\theta_1]^k$ in each universe X_i^k (see figure 4), we must verify that it is actually schedulable.

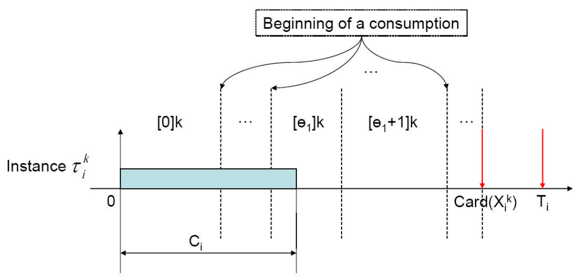

Figure 4. task τ_i potentially schedulable

We initialize

$$\begin{cases} C_i^{k,1} = C_i \\ B^{k,1} = C_i \\ q^{k,1} = \theta_1 \\ A^{k,1} = \sum_{m=0}^{\theta_1 - 1} card([m]^k) \\ r_{k,1} = C_i^{k,1} - A^{k,1} \end{cases}$$

For $l \geq 1$, we compute

$$B^{k,l+1} = \sum_{j=1}^{l} A^{k,j} + (r_{k,l} + \theta_l \cdot \alpha) \tag{17}$$

By using the same idea as that of a fixed-point algorithm, this computation stops as soon as either two consecutive values of $B^{k,j}, j \geq 1$, belong to the same cell or there exists $\mu_1 \geq 1$ such that $B^{k,\mu_1} > card(X_i^k)$. In the latter case, task τ_i is not schedulable because the deadline has been exceeded.
Actually, if $B^{k,l+1} \leq card(X_i^k)$, then $\exists \theta_{l+1} \geq 0$ such that

$$B^{k,l+1} \in [\theta_1 + \cdots + \theta_{l+1}]^k$$

If $\theta_{l+1} = 0$ then $B^{k,l+1}$ and $B^{k,l}$ belong to the same cell, then expression (18) holds with $\mu_2 = l + 1$ and $N_p(\tau_i^k)$ is given by (19), else if $\theta_{l+1} \neq 0$, we compute

$$\begin{cases} C_i^{k,l+1} = r_{k,l} + \theta_l \cdot \alpha \\ q^{k,l+1} = \theta_{l+1} \\ A^{k,l+1} = \sum_{m=\theta_1+\cdots+\theta_l}^{\theta_1+\cdots+\theta_{l+1}-1} card([m]^k) \\ r_{k,l+1} = C_i^{k,l+1} - A^{k,l+1} \end{cases}$$

and we compute the next value of $B^{k,j}$.
The algorithm is stopped as soon as

$$\exists \mu_2 \geq 1 \quad \text{such that} \quad \theta_{\mu_2} = 0 \tag{18}$$

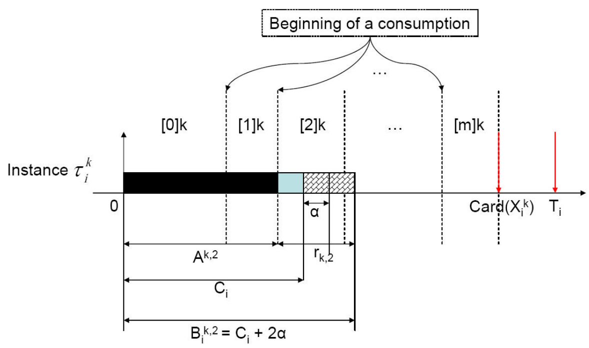

Figure 5. Computation of C_i^k

and therefore

$$N_p(\tau_i^k) = \sum_{j=1}^{\mu_2 - 1} q^{k,j} \tag{19}$$

For each $k = 1, \cdots, \sigma_i$, the PET C_i^k of task τ_i in X_i^k is given by

$$C_i^k = C_i + N_p(\tau_i^k) \cdot \alpha \qquad (20)$$

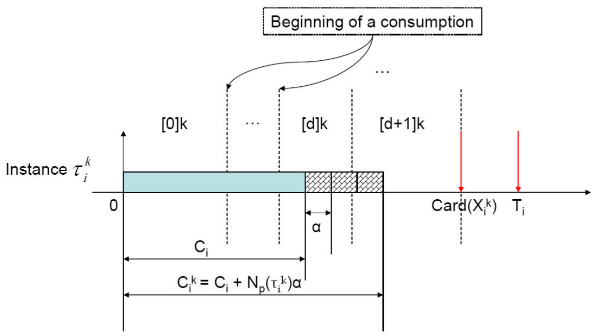

Figure 6. PET of task τ_i in instance τ_i^k: C_i^k

7: Deduce the image $\tau_i' = \pi(\tau_i) = \left(\left(C_i^1, C_i^2, \cdots, C_i^{\sigma_i}\right), T_i\right)$ of task τ_i resulting from the previous step.

8: Determine the response time $R_i^k, 1 \leq k \leq \sigma_i$ of task τ_i in its k^{th} instance, i.e. in the k^{th} T_i-*mesoid*. This is obtained by summing C_i^k and all the consumptions prior to C_i^k in the corresponding mesoid. Deduce the worst-case response time R_i of task τ_i.

$$R_i = max_{\{1 \leq k \leq \sigma_i\}}(R_i^k)$$

It is worth noticing that task τ_i is schedulable if and only if

$$R_i \leq T_i.$$

9: Build the sequence $\mathcal{L}_i^a$ by updating all the T_i-mesoids of the sequence $\mathcal{L}_i^b$.

10: Compute the *exact total utilization load factor at level i*. That is to say

$$U_i^* = \sum_{j=1}^{i} \frac{1}{\sigma_j} \left(\sum_{k=1}^{\sigma_j} \frac{C_j^k}{T_j} \right) = U_i + \sum_{j=1}^{i} \frac{1}{\sigma_j} \left(\sum_{k=1}^{\sigma_j} \frac{N_p(\tau_j^k) \cdot \alpha}{T_j} \right).$$

11: If $U_i^* \leq 1$ then increment i, and go back to step 2 as long as there remain potentially schedulable tasks in the system.

12: If $U_i^* > 1$, then the sub-system $\{\tau_j = (C_j, T_j)\}_{1 \leq j \leq i-1}$ was already maximal. In this case, the system $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq n}$ is not schedulable.

13: **end for**

Thanks to the above algorithm, a necessary and sufficient schedulability condition for a system of n tasks $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq n}$, all released at the same time and scheduled according to RMA, which takes the cost due to preemption accurately into account is given by

$$U_n^* = \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_n + \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \leq 1 \qquad (21)$$

Example

Still with the same assumption that $\alpha = 1$ let us consider $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ to be a system of four tasks with the characteristics defined in table 2.

Table 2. Characteristics of example 4

	C_i	T_i
τ_1	2	6
τ_2	3	10
τ_3	2	15
τ_4	3	30

According to RMA, the lower the index of a task is, the higher its priority is. Thus, as depicted in table 2, τ_1 has the highest priority and task τ_4 has the lowest priority. Thanks to our scheduling algorithm,

$\sigma_1 = 1$, thus for task τ_1

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \left\{ \mathcal{M}_1^{b,1} \right\} = \{\{\overbrace{1,2,3,4,5,6}^{[0]^1}\}\} \\ \tau_1 = (2,6) \longmapsto \tau_1' = ((2),6) \\ \mathcal{L}_1^a = \left\{ \mathcal{M}_1^{a,1} \right\} = \{\{(2),1,2,3,4\}\} \end{cases}$$

$\sigma_2 = 3$, thus for task τ_2

$$\tau_2 : \begin{cases} \mathcal{L}_2^b & = & \left\{ \mathcal{M}_2^{b,1}, \mathcal{M}_2^{b,2}, \mathcal{M}_2^{b,3} \right\} \\ & = & \{\{(2), \overbrace{1,2,3,4}^{[0]^1}, (2), \overbrace{5,6}^{[1]^1}\}, \\ & & \{\overbrace{1,2}^{[0]^2}, (2), \overbrace{3,4,5,6}^{[1]^2}, (2)\}, \\ & & \{\overbrace{1,2,3,4}^{[0]^3}, (2), \overbrace{5,6,7,8}^{[1]^3}\}\} \\ \tau_2 = (3,10) & \longmapsto & \tau_2' = ((3,4,3),6) \\ \mathcal{L}_2^a & = & \left\{ \mathcal{M}_2^{a,1}, \mathcal{M}_2^{a,2}, \mathcal{M}_2^{a,3} \right\} \\ & = & \{\{(5), 1, (2), 2, 3\}, \{(6), 1, 2, (2)\}, \\ & & \{(3), 1, (2), 2, 3, 4, 5\}\} \end{cases}$$

$\sigma_3 = 2$, thus for task τ_3

$$\tau_3 : \begin{cases} \mathcal{L}_3^b & = & \left\{ \mathcal{M}_3^{b,1}, \mathcal{M}_3^{b,2} \right\} \\ & = & \{\{(5), \overbrace{1}^{[0]^1}, (2), \overbrace{2,3}^{[1]^1}, (5)\}, \\ & & \{(1), \overbrace{1,2}^{[0]^2}, (5), \overbrace{3}^{[1]^2}, (2), \overbrace{4,5,6,7}^{[2]^2}\} \\ \tau_3 = (2,15) & \longmapsto & \tau_3' = ((3,2),15) \\ \mathcal{L}_3^a & = & \left\{ \mathcal{M}_3^{a,1}, \mathcal{M}_3^{a,2} \right\} \\ & = & \{\{(15)\}, \{(8), 1, (2), 2, 3, 4, 5\}\} \end{cases}$$

$\sigma_4 = 1$, thus for task τ_4

$$\tau_4 : \begin{cases} \mathcal{L}_4^b = \left\{ \mathcal{M}_4^{b,1} \right\} = \{\{(23), \overbrace{1}^{[0]^1}, (2), \overbrace{2,3,4,5}^{[1]^1}\}\} \\ \tau_4 = (3,30) \longmapsto \tau_4' = ((4),30) \\ \mathcal{L}_4^a = \left\{ \mathcal{M}_4^{a,1} \right\} = \{\{(29), 1\}\} \end{cases}$$

Therefore

$$U_4^* = \frac{2}{6} + \frac{1}{3}\left(\frac{3+4+3}{10}\right) + \frac{1}{2}\left(\frac{3+2}{15}\right) + \frac{4}{30} = 0.967$$

whereas

$$U_4 = \frac{2}{6} + \frac{3}{10} + \frac{2}{15} + \frac{3}{30} = 0.867$$

The schedule of this task set following RMA with the cost of preemption considered is depicted in figure 7.

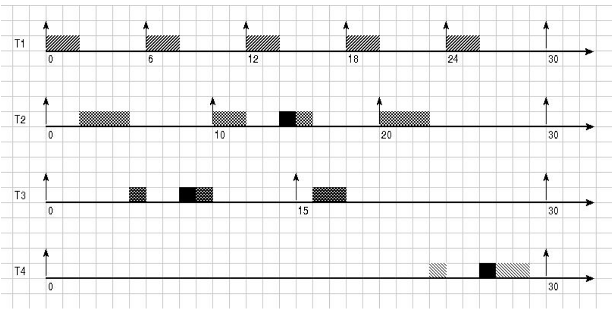

Figure 7. Execution of a task set by following RMA and considering the exact cost of preemption

whereas the schedule of the same set of tasks following RMA without the cost of preemption is depicted in figure 8.

6 Complexity of the proposed scheduling algorithm

For every task, the schedulability analysis is performed only once. In this analysis we walk through the iteration space in order to calculate the number and positions of available time units. Hence, the time and space complexity for every task is $O(m_0)$, where m_0 is the number of time units in the sequence of mesoids of each task.

To calculate the PET of a task τ_i which includes the exact cost of preemption within a given instance τ_i^k, the complexity of our algorithm is $O\left(\sigma_i \cdot \sigma_{hp} \cdot m_0\right)$, where σ_i is the number of instances of the current task in a hyperperiod at level i, and $\sigma_{hp} = \sum_{j=1}^{i-1} \sigma_j$ is the number of higher priority instances in a hyperperiod at level i. This complexity is explained as follows. Our analysis is a per instance analysis, and hence includes the factor σ_i for every task. For each instance, we need to calculate the exact number of preemptions and the PET which includes the exact cost of these preemptions. To calculate the exact number of preemptions, we need to partition every instance. Since the number of potential preemption occurences is equal to the number of higher priority instances within the hyperperiod at level i, the factor σ_{hp} is included. Although the complexity of the calculation of a PET adds a factor m_0 to the complexity, it is actually a small number since the range of available time units or availabilities between two consecutive consumptions is limited by the largest one. However, it is worth noticing that when the periods of the tasks form an harmonic sequence, the time and space complexity of our algorithm is $O(n)$, where n is the number of tasks in the task set.

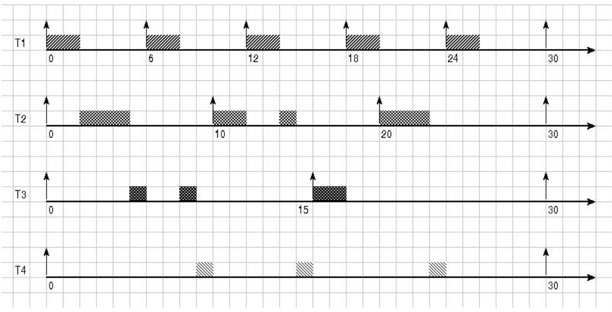

Figure 8. Execution of a task set by following RMA with the cost of preemption approximated

7 Conclusion and future work

This paper makes three main contributions. First, we give a counterexample on the critical instant when the cost of preemption is considered. Second, we present a technique which counts the exact number of preemptions for every intance of the task under consideration in a given task set. Finally, we provide an RMA extension which takes into account the exact cost due to preemption in the schedulability analysis rather than using an approximation. This technique provides a new and stronger schedulabily condition since no margins are necessary. Furthermore, this new condition always guarantees a correct execution of the system and eliminates the waste of resources.

Future work will study the case where the deadline of a task is smaller than its period.

References

[1] Joseph Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 1980.

[2] Ray Obenza and Geoff. Mendal. Guaranteeing real time performance using rma. *The Embedded Systems Conference, San Jose, CA*, 1998.

[3] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.

[4] Lehoczky-J.P. Sha, L. and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. *Proceedings of the IEEE Real-Time Systems Symposium*, 1986.

[5] J.P. Lehoczky, L. Sha, and Y Ding. The rate monotonic sheduling algorithm: exact characterization and average case bahavior. *Proceedings of the IEEE Real-Time Systems Symposium*, 1989.

[6] Lichen Zhang. Predictable architecture for real-time systems. *International Conference on Information, Communications and Signal Processing*, 1997.

[7] Engblom-J. Berg, C. and R. Wilhelm. Perspective workshop: Design of systems with predictable behaviour. *Dagstuhl Seminars*, 2004.

[8] H. Chetto and M. Chetto. On the acceptation of non-periodic time critical tasks in distributed systems. *Proc. 7th IFAC Workshop Distributed Computer Control Systems (DCCP-86)*, 1986.

[9] H. Chetto and M. Chetto. Some results on the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, 1989.

[10] H. Chetto, Silly M., and Bouchentouf T. Dynamic scheduling of real-time tasks under precedence contraints. *The Journal of Real-Time Systems*, 1990.

[11] Tindell K. Burns A. and Wellings A. Effective analysis for engineering real-time fixed priority schedulers. *IEEE Trans. Software Eng.*, 1995.

[12] I. Ripoll J. Echage and A. Crespo. Hard real-time preemptively scheduling with high context switch cost. *In Proceedings of the 7th Euromicro Workshop on Real-Time Systems*, 1995.

[13] Klein Mark H. Sha, Lui and John B. Goodenough. Rate monotonic analysis. *Technical Report CMU/SEI-91-TR-6 ESD-91-TR-6*, 1991.

[14] Tei-Wei Kuo Jian-Jia Chen. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. *LCTES'06, Ottawa, Ontario, Canada*, 2006.

[15] Buttazzo G. Bini, E. A hyperbolic bound for the rate monotonic algorithm. *IEEE Proc. of the 13th Euromicro Conf. on Real-Time Systems, pp. 59-66*, 2001.

[16] J. Goossens and Richard P. Overview of real-time scheduling problems. *Euro Workshop on Project Management and Scheduling, Invited Paper*, 2004.

[17] Giorgio C. Buttazzo. Rate monotonic vs edf: Judgement day. *Real-Time Systems, vol. 29, Number 1, pp. 5-26*, 2005.

[18] Mok A.K. Chen, D. and T. Kuo. Utilization bound revisited. *IEEE Transactions on computer, Vol. 52, No. 3, pp. 351-361*, 2003.

[19] Natarajan S. Park, D.W. and A. Kanevsky. Fixed priority scheduling of real-time systems using utization bounds. *Journal of Systems and Software, Elsevier. Vol. 33, pp.57-63*, 1996.

[20] M. Joseph and P. Pandya. Finding response times in real-time system. *BCS Computer Journal*, 1986.

[21] N.C. Audsley, A. Burns, M.F. Richardson, Tindell K., and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineerung Journal*, 1993.