Schedulability analysis for a combination of non-preemptive strict periodic tasks and preemptive sporadic tasks
Mohamed Marouf, Laurent George, Yves Sorel

To cite this version:
Mohamed Marouf, Laurent George, Yves Sorel. Schedulability analysis for a combination of non-preemptive strict periodic tasks and preemptive sporadic tasks. ETFA’12 - 17th IEEE International Conference on Emerging Technologies and Factory Automation, Sep 2012, Kraków, Poland. hal-00737917

HAL Id: hal-00737917
https://inria.hal.science/hal-00737917
Submitted on 3 Oct 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.
Schedulability analysis for a combination of non-preemptive strict periodic tasks and preemptive sporadic tasks

Mohamed MAROUF, Laurent GEORGE, Yves SOREL
INRIA Paris-Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: {Mohamed.Marouf, Laurent.George, Yves.Sorel}@inria.fr

Abstract

We consider the problem of fixed priority scheduling of non-preemptive strict periodic tasks in conjunction with sporadic preemptive tasks. There are few studies about the scheduling problem combining these two kinds of tasks. Moreover, only few results are available on scheduling non-preemptive strict periodic tasks since their performance analysis gives low success ratios, except in the case of harmonic tasks. Also, strict periodic tasks are of great importance since they are in charge for example of sensors/actuators or feedback control functions which are all critical in feedback control systems. Such tasks must have the highest priorities in order to guarantee a correct behavior of the control system. Preemptive sporadic tasks can be used for non critical functions and have lower priorities.

We first investigate the scheduling problem of non-preemptive strict periodic tasks by recalling an existing schedulability condition. This results in defining the first release times of strict periodic tasks that preserves the strict periodicity constraints. We show that the schedule of strict periodic tasks can have transient and permanent phases. Then, assuming that some non-preemptive strict periodic tasks have been scheduled, we characterize the release times of the sporadic tasks that maximize their worst case response times. We prove that these release times can be restricted to the permanent phase. For preemptive sporadic tasks, we extend the classical worst case response time computation to take into account non-preemptive strict periodic tasks. Finally, we consider the particular case where some of the sporadic tasks are alternate tasks to primary strict periodic tasks for fault-tolerance.

1 Introduction

We consider the problem of scheduling non-preemptive strict periodic tasks combined with preemptive sporadic tasks.

Strict periodic tasks are typically in charge of sensor/actuator or feedback control functions. The freshness of the information they use and/or the reactivity of the system are constrained. Indeed, for such tasks it is important to control their jitters (the difference between the worst case and the minimum response times) to ensure the system’s stability [17, 14, 3, 1].

We consider the Fixed Priority (FP) scheduling. We assume that all non-preemptive strict periodic tasks have the same highest fixed priority, whereas preemptive sporadic tasks have lower (distinct) fixed priority than strict periodic tasks. Afterwards, for the sake of clarity, we shall use the term ”strict periodic tasks” for ”non-preemptive strict periodic tasks”, and ”sporadic tasks” for ”preemptive sporadic tasks”.

In section 2, we first recall related work for strict periodic and sporadic tasks scheduling, and for fault-tolerant scheduling, then, we give the non-preemptive strict periodic task model. In section 3, we recall a necessary and sufficient schedulability condition for non-preemptive strict periodic tasks, and investigate the transient and permanent phases for such tasks. In section 4 we show how to determine the worst case scenario for a sporadic task where its Worst Case Response Time (WCRT) can be obtained, in the presence of strict periodic tasks. We prove that these release times belong only to the permanent phase of strict periodic tasks, and thus that the schedulability analysis for sporadic tasks can be restricted to the permanent phase. For preemptive sporadic tasks, we extend the classical necessary and sufficient schedulability condition based on the worst case response time computation [11] to take into account non-preemptive strict periodic tasks. Finally, in section 5, we consider fault-tolerance in the particular case where each primary strict periodic task has an alternate sporadic task which is released when its primary task fails.
2 System models and Related work

2.1 Related work

Preemptive scheduling has received considerable attention in the real-time community. However, we can notice that non-preemptive scheduling has received less attention from the real-time community.

Yet, non-preemptive scheduling problems should not be ignored since their resolutions may have great advantages in terms of schedulability. On the other hand, these problems are NP-Hard in the strong sense as Jeffay, Stanat and Martel [10] showed.

Baruah and Chakraborty [2] analyzed the schedulability of the non-preemptive recurring task model and showed that there exists polynomial time approximation algorithms for both preemptive and non-preemptive scheduling. Buttazzo and Cervin [5] used the non-preemptive task model to reduce the jitter of the tasks. A comprehensive schedulability analysis of non-preemptive systems was performed by George, Rivierre, and Spuri in [7]. The main difference between these works and the works proposed in this paper lies in the type of periods we consider for strict periodic tasks, i.e. strict periods. In the classical periodic model, the difference between the start times of two tasks jobs may vary whereas it must be a constant for strict periodic tasks.

Korst et al. proved in [13] a necessary and sufficient schedulability condition for two tasks which can be generalized for more than two tasks. In [6] Eisenbrand et al. proposed scheduling algorithms in the case of harmonic and non-harmonic strict periodic tasks. We proposed in [15, 16] a schedulability analysis for such tasks.

Software fault-tolerance has been considered through the primary/alternate task models. When a primary task cannot meet its deadline, an alternate task is run. The alternate task can be the same task (the task is re-executed). In [4], Burns et al. give a feasibility condition for sporadic tasks in the case of fixed priority scheduling. Faults are assumed to be detected at the completion time of the tasks. A task can only affect one task at a time. An alternate task can be the same task (the task is re-executed). In [8], Ghosh et al. consider a recovery mechanism to re-execute faulty tasks. They propose to use the available slack time to re-execute a faulty task in the case of RM scheduling. In [9], Han et al. consider software faults for primary tasks in the case of periodic tasks scheduled with RM. Primary tasks are more complex functions whose correctness is deemed more difficult to check.

2.2 Systems models

A non-preemptive strict periodic task \( \tau_i \) is denoted \( \tau_i(S^0_i, C_i, D_i, T_i) \), where:

- \( S^0_i \) is the first release time of \( \tau_i \)
- \( C_i \) the WCET (Worst Case Execution Time) of \( \tau_i \)
- \( D_i \) is the relative deadline of \( \tau_i \)

The start time of the \( k^{th} \) job of a strict periodic task \( \tau_i \) is given by \( S^k_i = S^0_i + k \cdot T_i \). Figure 1 shows an example of a strict periodic task.

![Figure 1. Model for non-preemptive strict periodic task \( \tau_i(S^0_i, C_i, D_i, T_i) \)](image)

A preemptive sporadic task is non-concrete, i.e. its first release time can be chosen arbitrarily. A preemptive sporadic task \( \tau_i \) is therefore denoted (w.r.t. the strict periodic task model) \( \tau_i(C_i, D_i, T_i) \). \( T_i \) is the minimum inter-arrival time between two successive jobs of \( \tau_i \). We consider the case \( D_i \leq T_i \) for sporadic tasks.

The scheduling used in this paper is Fixed Priority (FP) scheduling. Tasks in \( \Gamma^S \) have the same highest priority while all tasks in \( \Gamma^{NS} \) have distinct lower priority tasks. Furthermore, we assume that task parameters are integers multiple of the tick time.

3 Scheduling strict periodic tasks

3.1 Necessary and sufficient schedulability condition

For the sake of clarity, we use this notification \( g_{i,j} \) for the gcd (Greatest Common Divisor) of two periods \( T_i, T_j \):

\[ g_{i,j} = \gcd(T_i, T_j). \]

Korst et al. gave in [12] a necessary and sufficient schedulability condition for a task set. It consists of applying a necessary and sufficient condition valid for only two tasks to all possible pairs of tasks.

Theorem 1 A task set \( \Gamma^S = \{ \tau_i(S^0_i, C_i, D_i, T_i), i = 1, n \} \) is schedulable if and only if for all pairs of tasks \( (\tau_i, \tau_j) \) satisfy the condition

\[ C_i \leq (S^0_i - S^0_j) \mod g_{i,j} \leq g_{i,j} - C_j \tag{1} \]

Now we characterize the transient and the permanent phases of the schedule considering only strict periodic tasks.

3.2 Transient and permanent phases

The transient phase \( \Phi = [0, \phi] \) ends at a time \( \phi \geq 0 \), which is the smallest time such that the release times obtained in any time interval \( [\phi + kL, \phi + (k + 1)L] \), \( k \in \mathbb{N}^+ \), relatively to \( \phi + kL \), are the same as those obtained in the time interval \( [\phi, \phi + L] \), relatively to \( \phi \). By definition, \( L \) is the length of the permanent phase.
The following theorem gives the time interval corresponding to transient phase obtained for the schedule of strict periodic tasks.

**Theorem 2** We consider a schedule of strict periodic tasks. The transient phase $\Phi$ of this schedule is the time interval given by: $\Phi = [0, \phi]$, where

$$\phi = \max(0, \max_{i=1..n} \{S_i^0 + C_i - T_i\}) \quad (2)$$

**Proof**

\[ \phi \] is the smallest integer such as the first permanent phase $[\phi, \phi + L]$ contains $\left(\frac{\phi}{T_i}\right)$ jobs of each task $\tau_i$. Thus the $(\frac{L}{T_i} - 1)^{th}$ job of $\tau_i$ must end its execution before $(\phi + L)$, thus

$$S_i^0 + \left(\frac{L}{T_i} - 1\right)T_i + C_i \leq \phi + L$$

thus

$$\phi \geq S_i^0 + C_i - T_i. \quad (3)$$

Furthermore, for any task $\tau_i$, its relative start time according to $\phi$ must be equal to its relative start time according to $\phi + L$.

The relative start time according to $\phi$ is given by

$$S_i^0 + \left[\frac{\phi - s_i}{T_i}\right] T_i - \phi.$$  

The relative start time according to $\phi + L$ is given by

$$S_i^0 + \left[\frac{(L + \phi) - s_i}{T_i}\right] T_i - (L + \phi) = S_i^0 + \left[\frac{\phi - s_i}{T_i}\right] T_i + L - L - \phi = S_i^0 + \left[\frac{\phi - s_i}{T_i}\right] T_i - \phi.$$  

These two relative start times are thus equal.  

As $\phi$ is the smallest integer which satisfies condition $(3)$ then

$$\phi = \max_{i=1..n} \{0, s_i + C_i - T_i\}. \quad (3)$$

\[ \square \]

## 4 Combining strict periodic and sporadic tasks

In this section, we study the schedulability of a combination of strict periodic and sporadic tasks. We assume that a set of non-preemptive strict periodic tasks have already been scheduled with the highest priority, and a set of preemptive sporadic tasks is to be scheduled with lower (but different) priorities.

We use the following notations:

- $\Gamma^S$ (resp. $\Gamma^NS$) denotes the task set corresponding to strict periodic (respectively sporadic) tasks.
- $hp^NS(i)$ denotes the set of tasks in $\Gamma^NS$ having higher priority than a task $\tau_i$ in $\Gamma^NS$.

### 4.1 Schedulability analysis for sporadic tasks

In order to study the schedulability of sporadic tasks when tasks with strict periods have been scheduled, we have to determine the critical instants for a sporadic task where the WCRT can be reached.

It has been proved in [11] that a critical instant occurs when the release time of a sporadic task is equal to the release time of all higher priority tasks. As strict periodic tasks have all higher priority than all the sporadic tasks, we define the set of critical instants $\Psi$ which contains all the start times of the strict periodic jobs in the transient and the permanent phases: $[0, \phi + L]$.

To consider tasks with strict periods, we have to study all cases of release times in $\Psi$ and compute the WCRT of a sporadic task $\tau_i$ when its first release time corresponds to the release time of a strict periodic task. For a sporadic task $\tau_i$, this results in first releasing, at the release time of a strict periodic task in $\Psi$ all higher priority sporadic tasks than $\tau_i$ at the same time. However, rather than testing all the release times of the transient phase in $\Psi$, some release times are useless and will be thus removed from $\Psi$ according to lemma 1.

**Lemma 1** Consider a sequence of consecutive executions of strict periodic tasks with no slack between the executions. Then only the release time of the first strict periodic task in the sequence should be considered for the computation of worst case response time of a sporadic task.

**Proof** Consider a sequence $seq$ of consecutive executions of strict periodic tasks corresponding to a subset of tasks in $\Gamma^S$. Let 0 be the time origin corresponding to the release time of the first strict periodic tasks in the sequence $seq$. As sporadic tasks have smaller priority than strict periodic tasks, a sporadic task released at time $t_i \geq 0$ can start its execution only when all strict periodic tasks in $seq$ are completed and will terminate at the same time for any release time $t_i$ belonging to $[0, duration(seq)]$, where $duration(seq)$ is the sum of the execution times of all strict periodic tasks in $seq$. The response time of a sporadic task is therefore maximum when a sporadic task is released at time 0. Hence the lemma.  

Thus, if $\exists S_k^i, S_j^i \in \Psi$ such as $S_k^i - S_j^i = C_j$, then $S_k^i$ is removed from $\Psi$.

Finally, we limit the release times in $\Psi$ only to the release times of strict periodic tasks in the permanent phase according to lemma 2.

**Lemma 2** In the set $\Psi$, only the release times which belong to the permanent phase of strict periodic tasks should be considered to study the WCRT of sporadic tasks.

**Proof** For strict periodic tasks, the time interval $[L, \phi + L]$ in the permanent phase contains the release time pattern corresponding to all the release times of the transient phase.
The transient phase may contain less jobs than in the permanent phase. Thus, jobs of the permanent phase are more critical than those of the transient phase for sporadic tasks.

The following theorem gives the computational requirements at time t for a sporadic task \( \tau_i \) released at time \( S \in \Psi \).

**Theorem 3** Consider a strict periodic task set \( \Gamma^S \) and a sporadic task set \( h_{p_{NS}}(i) \) already scheduled. Let \( \tau_i \) be a sporadic task released at time \( S \in \Psi \). The sum of the computational requirements at time t (w.r.t time S) are given by

\[
W_i(t) = C_i + \sum_{\tau_j \in h_{p_{NS}}(i)} \left[ \frac{t}{T_j} \right] C_j + \sum_{\tau_j \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j
\]

where \( s_j \) is the relative start time \( S_j^0 \) according to a release time \( S \) given by

\[
s_j = S_j^0 + \left[ \frac{S - S_j^0}{T_j} \right] T_j - S
\]

**Proof** Consider that a sporadic task \( \tau_i \) is first released at time \( S \in \Psi \). The sum of the computational requirements at time t (w.r.t time S) \( W_i(t) \) is the sum of the following computational requirements:

1. one execution of \( \tau_i \) starting at time \( S \): \( C_i \);
2. all strict periodic tasks (all with higher priority than \( \tau_i \)):
   \[
   \sum_{\tau_j \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j;
   \]
3. all sporadic tasks with higher priority than \( \tau_i \):
   \[
   \sum_{\tau_j \in h_{p_{NS}}(i)} \left[ \frac{t}{T_j} \right] C_j.
   \]

\( \square \)

The following theorem gives a schedulability necessary and sufficient condition for set of sporadic task.

**Theorem 4** Consider a strict periodic task set \( \Gamma^S \) already scheduled. A sporadic task set \( \Gamma^{NS} \) is schedulable if and only if

\[
\forall \tau_i \in \Gamma^{NS} : R_i \leq D_i
\]

where \( R_i \) is the WCRT of \( \tau_i \in \Gamma^{NS} \), which is the solution of \( R_i = W(R_i) \) computed by iteration.

**Proof** The proof is identical to the one given in [11] which states that the Worst Case Response Time of any task should be less than or equal to its deadline. \( \square \)

---

**Example**

We use the Rate-Monotonic algorithm to schedule the following tasks. Consider the strict periodic task set \( \Gamma^S = \{ \tau_1(0, 1, 4, 4), \tau_2(1, 1, 6, 6), \tau_3(2, 1, 12, 12) \} \) and the sporadic task set \( \Gamma^{NS} = \{ \tau_4(2, 6, 8), \tau_5(2, 12, 12) \} \) to be scheduled.

For \( (\tau_1, \tau_2), g_{1,2} = 2 \) and condition (1) is satisfied:

\[
1 \leq (1 - 0) \mod 2 \leq 2 - 1 \implies 1 \leq 1 \leq 1.
\]

For \( (\tau_1, \tau_3), g_{1,3} = 4 \) and condition (1) is satisfied:

\[
1 \leq (2 - 0) \mod 4 \leq 4 - 1 \implies 1 \leq 2 \leq 3.
\]

For \( (\tau_2, \tau_3), g_{2,3} = 6 \) and condition (1) is satisfied:

\[
1 \leq (2 - 1) \mod 6 \leq 6 - 1 \implies 1 \leq 1 \leq 5.
\]

Thus, \( \Gamma^S \) is schedulable.

According to theorem 2, the schedule of \( \Gamma^S \) has no transient phase. The permanent phase starts at time 0 and has a length \( L = \text{lcm}(T_1, T_2, T_3) = 12 \), thus \( \Psi = \{0, 1, 2, 4, 7, 8\} \) corresponding to the release times of tasks \( \Gamma^S \) in the time interval \([0, L]\). After pruning the release times of strict periodic tasks according to lemma 1 we have: \( \Psi = \{0, 4, 7\} \).

From theorem 3 we have:

\[
W_4(t) = 2 + \left[ \frac{t - s_1}{4} \right] + \left[ \frac{t - s_2}{6} \right] + \left[ \frac{t - s_3}{12} \right],
\]

and

\[
W_5(t) = 2 + \left[ \frac{t}{8} \right] 2 + \left[ \frac{t - s_1}{4} \right] + \left[ \frac{t - s_2}{6} \right] + \left[ \frac{t - s_3}{12} \right].
\]

According to theorem 4, the response times of \( \tau_4 \) and \( \tau_5 \) for the release times in \( S \in \Psi \) are given by:

\[
\begin{array}{cccc|cc}
S & s_1 & s_2 & s_3 & R_4 & R_5 \\
0 & 0 & 1 & 2 & 6 & 12 \\
4 & 0 & 3 & 4 & 3 & 7 \\
7 & 1 & 0 & 7 & 4 & 9 \\
\end{array}
\]

As \( R_4 \leq D_4 \) and \( R_5 \leq D_5 \), \( \tau_4 \) and \( \tau_5 \) are schedulable as shown in figure 2.

---

**Figure 2. Scheduling diagram of strict periodic and sporadic tasks**
5 Extension to fault-tolerance

We consider a task set of non-preemptive strict periodic tasks (control tasks, sensors/actuators tasks, etc.) and preemptive sporadic tasks (asynchronous events outside the embedded computer, etc.). For reliable systems, we proposed to use an alternate sporadic tasks for the primary strict periodic tasks, in such a way that when a primary non-preemptive strict periodic task fails, an alternate preemptive sporadic task is released. We suppose that faults occurs during the execution of primary tasks, and alternate tasks must meet the absolute deadlines of their primary task.

In this section we consider that sporadic tasks are composed of alternate tasks and independent sporadic tasks.

Primary strict periodic tasks have higher priority than alternate tasks that have higher priority than independent sporadic tasks. Both sporadic task sets are scheduled using RM or DM algorithms.

Figure 3 shows a primary task with three faults occurring: during the execution of the first job, during the execution of the third job and at the beginning of the execution of the forth job.

![Figure 3. A primary strict periodic task with its alternate sporadic tasks](image)

The worst case which minimizes the relative deadline of an alternate job is obtained when the fault of its primary job occurs at the end of its execution. As shown in figure 3, the minimal relative deadline $D_{min}$ and minimal inter-arrival time $T_{min}$ are obtained when a fault occurs at the the end of execution of a job and the next fault occurs at the release time of the next job. In that case we have: $D_{min} = D_i - C_i$ and $T_{min} = T_i - C_i$. For this reason we consider that for each primary task $\tau_i(S^i_0, C_i, D_i, T_i)$, the alternate task, denoted $\tau_{i,a}$ has an execution time $C_{i,a} \leq D_i - C_i$, a relative deadline $D_{i,a} = D_i - C_i$, and a minimal inter-arrival time $(T_i - C_i)$. It is given by $\tau_i(S^0_{i,a}, C_{i,a}, (D_i - C_i), (T_i - C_i))$, with $S^0_i \leq S^0_{i,a} \leq S^0_i + C_i$.

### 5.1 Schedulability analysis for alternate tasks

In this section we study the schedulability of a combination of primary strict periodic tasks and their alternate sporadic tasks.

We use the following notations:

- $\Gamma^S$ (resp. $\Gamma^A$) denotes the task set corresponding to primary strict periodic (resp. alternate sporadic) tasks.
- $h\pi_a^S(i)$ denotes the set of tasks in $\Gamma^A$ having higher priority than a task $\tau_{i,a}$ in $\Gamma^S$.

In order to consider the worst case response time for an alternate task, we consider that all its primary jobs fails at the end of their executions, which mean that each primary job is entirely executed before releasing the alternate job. Thus, the critical instants for each alternate task $\tau_{i,a}$ are obtained for $t = S^k_i + C_i$, where $S^k_i$ belongs to the permanent phase. Thus, the critical instants of each alternate task $\tau_{i,a}$ is given by $\Psi_i = \{ S^k_i + C_i, \phi \leq S^k_i < \phi + L \}$.

In order to compute the worst case response time of an alternate job $\tau_{i,a}^k$, we introduce the following theorem which gives the computational requirements at time $t$ for an alternate task $\tau_{i,a}$ released at time $t \in \Psi_i$.

**Theorem 5** Let $\tau_{i,a}$ be an alternate task of a primary task $\tau_i$. $\tau_{i,a}$ is released at time $t \in \Psi_i$. Let $\Gamma^S$ be a set of primary strict periodic tasks already scheduled. Let $h\pi_a^S(i)$ be a set of alternate tasks having higher priorities than $\tau_{i,a}$ already scheduled. The sum of the computational requirements at time $t \geq 0$ (w.r.t time $S$) are given by

$$W_{i,a}(t) = C_{i,a} + \sum_{t \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j + \sum_{t \in h\pi_a^S(i)} \left[ \frac{t - s_j}{T_j} \right] C_{j,a} + \sum_{t \in \Gamma^S} \max [0, (R_{i,a}(S + s_j - T_j) + (s_j - T_j))]$$

where $s_j$ is the relative start time $S^k_i$ of $\tau_i$ according to a release time $S$ given by

$$s_j = S^0_j + \left[ \frac{S - S^0_j}{T_j} \right] T_j - S$$

**Proof**

Consider the alternate task $\tau_{i,a}$ released at time $t \in \Psi_i$. The sum of the computational requirements at time $t \geq 0$ (w.r.t time $S$) $W_{i,a}(t)$ is the sum of the following computational requirements:

1. one execution of $\tau_{i,a}$ released at time $S$: $C_i$;
2. strict periodic tasks (all with higher priority than $\tau_{i,a}$):

$$\sum_{t \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j;$$

3. alternate tasks with higher priorities than $\tau_{i,a}$:

$$\sum_{t \in h\pi_a^S(i)} \left[ \frac{t - s_j}{T_j} \right] C_{j,a}$$
4. the sum of the additional computational requirements of alternate jobs of $h_{p_{a}^{NS}}$ executed before time $S$ and which have not finished their executions yet before time $S$. Let $\tau_{j,a}^k$ be the last job of $\tau_{j,a}$ executed before time $S$. The start time of this job is $(S + S_j - T_j)$. Its response time calculated according to theorem 4 at its start time is $R_{i,a}(S + S_j - T_j)$. Thus, the additional computational requirements is given by:

$$R_{i,a}(S + S_j - T_j) - (S - (S + s_j - T_j)) = R_{i,a}(S + S_j - T_j) + (s_j - T_j).$$

If this job finishes its execution before time $S$ then the additional computational requirements is equal to zero. Thus for each task $\tau_{j,a} \in h_{p_{a}^{NS}}(i)$, the additional computational requirements is equal to:

$$\max [0, (R_{i,a}(S + s_j - T_j) + s_j - T_j)].$$

The sum of all the additional computational requirements is equal to:

$$\sum_{\tau_j \in h_{p_{a}^{NS}}(i)} \max [0, (R_{i,a}(S + s_j - T_j) + s_j - T_j)].$$

\[\square\]

### 5.2 Schedulability analysis for alternate and sporadic tasks

After scheduling primary and alternates tasks, we focus now on scheduling sporadic tasks which have the lowest priorities. The following theorem gives the computational requirements at time $t$ for a sporadic task $\tau_i$ released at time $S \in \Psi$.

**Theorem 6** Let $\tau_i$ be a sporadic task released at time $S \in \Psi$. Let $\Gamma^S$ (resp. $\Gamma_a^{NS}$) be the task set corresponding to primary strict periodic (resp. alternate sporadic) tasks already scheduled. The sum of the computational requirements at time $t \geq 0$ (w.r.t time $S$) are given by

$$W_i(t) = C_i + \sum_{\tau_j \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j + \sum_{\tau_j \in \Gamma_a^{NS}} \left[ \frac{t - s_j}{T_j} \right] C_{j,a} + \sum_{\tau_j \in h_{p_{a}^{NS}}(i)} \max [0, (R_{i,a}(S + s_j - T_j) + s_j - T_j)].$$

where $s_j$ is the relative start time $S_j^0$ of $\tau_i$ according to a release time $S$ given by

$$s_j = S_j^0 + \left[ \frac{S - S_j^0}{T_j} \right] T_j - S \quad (10)$$

**Proof**

Let us consider the sporadic task $\tau_i(C_i, D_i, T_i)$ released at time $S \in \Psi_i$. The sum of the computational requirements at time $t \geq 0$ (w.r.t time $S$) $W_i(t)$ is the sum of the following computational requirements:

1. one execution of $\tau_{i,a}$ released at time $S$: $C_i$
2. primary tasks of $\Gamma^S$: 

$$\sum_{\tau_j \in \Gamma^S} \left[ \frac{t - s_j}{T_j} \right] C_j;$$
3. alternate tasks of $\Gamma_a^{NS}$:

$$\sum_{\tau_j \in \Gamma_a^{NS}} \left[ \frac{t - s_j}{T_j} \right] C_{j,a};$$
4. the sum of the additional computational requirements of all alternate jobs of $\Gamma_a^{NS}$ executed before time $S$ and which have not finished their executions yet at time $S$:

$$\sum_{\tau_j \in h_{p_{a}^{NS}}(i)} \max [0, (R_{i,a}(S + s_j - T_j) + s_j - T_j)];$$
5. sporadic tasks with higher priorities than $\tau_i$ released at time $S$:

$$\sum_{\tau_j \in h_{p_{a}^{NS}}(i)} \left[ \frac{t}{T_j} \right] C_j.$$

\[\square\]

**Example**

Consider the following task sets to be scheduled: primary task set $\Gamma^S = \{\tau_1(0, 4, 9, 12), \tau_2(4, 2, 13, 18)\}$, alternate task set $\Gamma_a^{NS} = \{\tau_{1,a}(4, 5, 12), \tau_{2,a}(2, 11, 18)\}$ and sporadic task set $\Gamma^{NS} = \{\tau_3(4, 36, 36)\}$.

For $(\tau_1, \tau_2)$, $g_{1,2} = 6$ and condition (1) is satisfied:

$$4 \leq (4 - 0) \text{mod} 6 \leq 6 - 2 \implies 4 \leq 5 \leq 5.$$

Thus, $\Gamma^S$ is schedulable.

According to theorem 2, the schedule of $\Gamma^S$ has no transient phase, thus the permanent phase stars at time 0 and has a length $L = LCM(T_1, T_2) = 36$.

The release times of $\tau_1$ in the permanent phase are $\Psi_1' = \{0, 12, 24\}$, thus $\Psi_1 = \{4, 16, 28\}$.

The release times of $\tau_2$ in the permanent phase are $\Psi_2' = \{4, 22\}$ thus $\Psi_2 = \{6, 24\}$.

The critical instants of $\Gamma^S$ are: $\Psi = \Psi_1' \cup \Psi_2' = \{0, 4, 12, 22, 24\}$. After pruning the release times of strict periodic tasks according to lemma 1 we have: $\Psi = \{0, 12, 22\}$.

For $\tau_{1,a}$ we have:

$$W_{1,a}(t) = 4 + \left[ \frac{t - s_1}{T_1} \right] C_1 + \left[ \frac{t - s_2}{T_2} \right] C_2$$
Thus, we have:

\[ W_3 = 2 + \left[ \frac{t - s_1}{12} \right] 4 + \left[ \frac{t - s_2}{18} \right] 2 + \left[ \frac{t - s_{1,a}}{12} \right] 4 + \left[ \frac{t - s_{2,a}}{18} \right] 2 \]

The response times \( R_3 \) of \( \tau_3 \) are given by:

\[
\begin{array}{c|c|c|c|c|c}
S & s_1 & s_2 & s_{1,a} & s_{2,a} & R_3(S) \\
0 & 0 & 4 & 4 & 6 & 36 \\
12 & 0 & 10 & 4 & 12 & 24 \\
22 & 0 & 2 & 6 & 2 & 36 \\
\end{array}
\]

As \( R_3 \leq (D_3 = 36) \), \( \tau_3 \) is schedulable.

Figure 4 shows the scheduling diagram of primary, alternate and sporadic tasks within a permanent phase in the worst case failures occurrences. The first job of \( \tau_1,a \) and \( \tau_2,a \) and the second job of \( \tau_2,a \) meet their respective deadlines.

![Figure 4. Scheduling diagram within a permanent phase in the worst case failures occurrences](image)

Figure 5 shows the scheduling diagram of primary, alternate and sporadic tasks within a permanent phase in the case of arbitrary failures occurrences. All tasks responses times are less than their respective deadlines. For instance, the first job of \( \tau_1,a \) starts its execution before its primary job finishes its execution, and thus its response time is less than its deadline.

![Figure 5. Scheduling diagram within a permanent phase in the case of arbitrary failures occurrences](image)
6 Conclusion

In this paper we have considered the problem of scheduling, with fixed priorities, strict periodic tasks in conjunction with sporadic tasks. Strict periodic tasks have the highest priority. We first present a necessary and sufficient schedulability condition valid for strict periodic tasks. This results in defining the first release times of strict periodic tasks preserving the strict periodicity constraints. Then, we show that the schedule of strict periodic tasks can have a transient and a permanent phase. We characterize the duration on both phases and show that we only need to consider the permanent phase for the schedulability of sporadic tasks. The release time of sporadic tasks can be chosen arbitrarily. We show that a worst case response time analysis can be used for sporadic tasks. The worst case response time of a sporadic task is obtained by releasing all sporadic tasks synchronously, at the release time of a strict periodic task in the permanent phase. We also show how to prune the times to consider (corresponding to the release times of strict periodic tasks) in the permanent phase. We give the schedulability condition for sporadic tasks based on the worst case response time computation of the tasks. Finally, we extend these results to the case where some of the sporadic tasks are alternate tasks to primary periodic tasks for fault-tolerance, and we give the corresponding worst case response time calculation.

References