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 is run to re-execute the faulty task. 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 τ_i is denoted $\tau_i(S_i^0, C_i, D_i, T_i)$, where:

- S_i^0 is the first release time of τ_i
- C_i the WCET (Worst Case Execution Time) of τ_i
- D_i is the relative deadline of τ_i
- T_i is the strict period of τ_i.

The start time of the k^{th} job of a strict periodic task τ_i is given by $S_i^k = S_i^0 + k \cdot T_i$. Figure 1 shows an example of a strict periodic task.

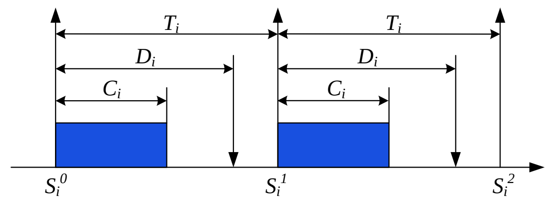

Figure 1. Model for non-preemptive strict periodic task $\tau_i(S_i^0, C_i, D_i, T_i)$

A preemptive sporadic task is non-concrete, i.e. its first release time can be chosen arbitrarily. A preemptive sporadic task τ_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 τ_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 Γ^S have the same highest priority while all tasks in Γ^{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* $\Gamma^S = \{\tau_i(S_i^0, C_i, D_i, T_i), i = 1, n\}$ *is schedulable if and only if for all pairs of tasks* (τ_i, τ_j) *satisfy the condition*

$$C_i \leq (S_j^0 - S_i^0) mod\, g_{i,j} \leq g_{i,j} - C_j \qquad (1)$$

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 ϕ. 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* Φ *of this schedule is the time interval given by:* $\Phi = [0, \phi[$*, where*

$$\phi = Max(0, \underset{i=1..n}{Max}\left\{S_i^0 + C_i - T_i\right\}) \tag{2}$$

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

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

thus

$$\phi \geq S_i^0 + C_i - T_i. \tag{3}$$

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

The relative start time according to ϕ is given by

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

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

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

These two relative start times are thus equal.

As ϕ is the smallest integer which satisfies condition (3) then

$$\phi = \underset{i=1,\ldots,n}{Max}(0, s_i + C_i - T_i).$$

□

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:

- Γ^S (resp. Γ^{NS}) denotes the task set corresponding to strict periodic (respectively sporadic) tasks.
- $hp^{NS}(i)$ denotes the set of tasks in Γ^{NS} having higher priority than a task τ_i in Γ^{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 tasks 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 Ψ 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 Ψ and compute the WCRT of a sporadic task τ_i when its first release time corresponds to the release time of a strict periodic task. For a sporadic task τ_i, this results in first releasing, at the release time of a strict periodic task in Ψ all higher priority sporadic tasks than τ_i at the same time. However, rather than testing all the release times of strict periodic tasks in Ψ, some release times are useless and will be thus removed from Ψ 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 Γ^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 $\exists S_i^k,\ S_j^l \in \Psi$ such as $S_i^k - S_j^l = C_j$, then S_i^k is removed from Ψ.

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

Lemma 2 *In the set* Ψ*, 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

$[0, \phi[$. 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. $\square$

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

Theorem 3 *Consider a strict periodic task set Γ^S and a sporadic task set $hp^{NS}(i)$ already scheduled. Let τ_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*

$$W_i(t) = C_i + \sum_{\tau_j \in hp^{NS}(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j + \sum_{\tau_j \in \Gamma^S} \left\lceil \frac{t - s_j}{T_j} \right\rceil C_j \tag{4}$$

where s_j is the relative start time S_j^k according to a release time S given by

$$s_j = S_j^0 + \left\lceil \frac{S - S_j^0}{T_j} \right\rceil T_j - S \tag{5}$$

Proof
Consider that a sporadic task τ_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 τ_i starting at time S: C_i;
2. all strict periodic tasks (all with higher priority than τ_i):
$$\sum_{\tau_j \in \Gamma^S} \left\lceil \frac{t - s_j}{T_j} \right\rceil C_j;$$
3. all sporadic tasks with higher priority than τ_i:
$$\sum_{\tau_j \in hp^{NS}(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j.$$

$\square$

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

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

$$\forall \tau_i \in \Gamma^{NS} : \; R_i \leq D_i \tag{6}$$

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 (τ_1, τ_2), $g_{1,2} = 2$ and condition (1) is satisfied:

$$1 \leq (1-0)mod2 \leq 2-1 \Longrightarrow 1 \leq 1 \leq 1.$$

For (τ_1, τ_3), $g_{1,3} = 4$ and condition (1) is satisfied:

$$1 \leq (2-0)mod4 \leq 4-1 \Longrightarrow 1 \leq 2 \leq 3.$$

For (τ_2, τ_3), $g_{2,3} = 6$ and condition (1) is satisfied:

$$1 \leq (2-1)mod6 \leq 6-1 \Longrightarrow 1 \leq 1 \leq 5.$$

Thus, Γ^S is schedulable.

According to theorem 2, the schedule of Γ^S has no transient phase. The permanent phase starts at time 0 and has a length $L = lcm(T_1, T_2, T_3) = 12$, thus $\Psi = \{0, 1, 2, 4, 7, 8\}$ corresponding to the release times of tasks Γ^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:

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

and

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

According to theorem 4, the response times of τ_4 and τ_5 for the release times in $S \in \Psi$ are given by:

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

As $R_4 \leq D_4$ and $R_5 \leq D_5$, τ_4 and τ_5 are schedulable as shown in figure 2.

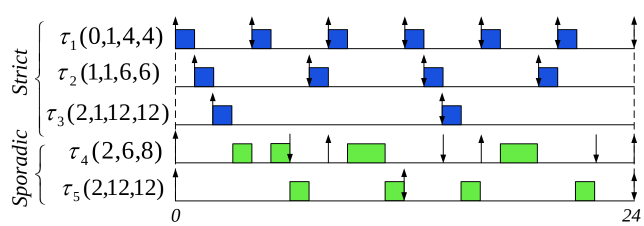

Figure 2. Scheduling diagram of strict periodic and sporadic tasks

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.

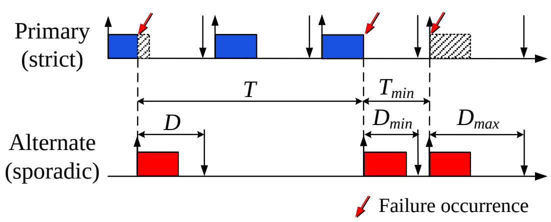

Figure 3. A primary strict periodic task with its alternate sporadic tasks

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 $\tau_i(S_{i,a}^0, C_{i,a}, (D_i - C_i), (T_i - C_i))$, with

$$S_i^0 \leq S_{i,a}^0 \leq S_i^0 + C_i.$$

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:

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

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 $t = S_i^k + C_i$, where S_i^k belongs to the permanent phase. Thus, the critical instants of each alternate task $\tau_{i,a}$ is given by $\Psi_i = \{S_i^k + C_i,\ \phi \leq S_i^k < \phi + L\}$.

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 $S \in \Psi_i$.

Theorem 5 *Let $\tau_{i,a}$ be an alternate task of a primary task τ_i. $\tau_{i,a}$ is released at time $S \in \Psi_i$. Let Γ^S be a set of primary strict periodic tasks already scheduled. Let $hp_a^{NS}(i)$ 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*

$$W_{i,a}(t) = C_{i,a}$$
$$+ \sum_{\tau_j \in \Gamma^S} \left\lceil \tfrac{t - s_j}{T_j} \right\rceil C_j + \sum_{\tau_j \in hp_a^{NS}(i)} \left\lceil \tfrac{t - s_j}{T_j} \right\rceil C_{j,a}$$
$$+ \sum_{\tau_j \in hp_a^{NS}(i)} max\left[0,\ (R_{i,a}(S + s_j - T_j) + (s_j - T_j))\right] \quad (7)$$

where s_j is the relative start time S_j^k of τ_i according to a release time S given by

$$s_j = S_j^0 + \left\lceil \frac{S - S_j^0}{T_j} \right\rceil T_j - S \quad (8)$$

Proof
Consider the alternate task $\tau_{i,a}$ released at time $S \in \Psi_i$. The sum of the computational requirements at time $t \geq 0$ (w.r.t time S) $W_{i,a}(t)$ 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 $\tau_{i,a}$):
$$\sum_{\tau_j \in \Gamma^S} \left\lceil \frac{t - s_j}{T_j} \right\rceil C_j;$$
3. alternate tasks with higher priorities than $\tau_{i,a}$:
$$\sum_{\tau_j \in hp_a^{NS}(i)} \left\lceil \frac{t - s_j}{T_j} \right\rceil C_{j,a}$$

4. the sum of the additional computational requirements of alternate jobs of hp_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 $\tau_{j,a} \in hp_a^{NS}(i)$, the additional computational requirements is equal to:

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

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

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

□

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 τ_i released at time $S \in \Psi$.

Theorem 6 *Let τ_i be a sporadic task released at time $S \in \Psi$. Let Γ^S (resp. Γ_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*

$$W_i(t) = C_i + \sum_{\tau_j \in \Gamma^S} \left\lceil \frac{t-s_j}{T_j} \right\rceil C_j$$

$$+\sum_{\tau_j \in \Gamma_a^{NS}} \left\lceil \frac{t-s_j}{T_j} \right\rceil C_{j,a} + \sum_{\tau_j \in hp^{NS}(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j \quad (9)$$

$$+\sum_{\tau_j \in \Gamma_a^{NS}} max\left[0,\ (R_i(S+s_j-T_j)+s_j-T_j)\right]$$

where s_j is the relative start time S_j^k of τ_i according to a release time S given by

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

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 Γ^S:
$$\sum_{\tau_j \in \Gamma^S} \left\lceil \frac{t-s_j}{T_j} \right\rceil C_j;$$
3. alternate tasks of Γ_a^{NS}:
$$\sum_{\tau_j \in \Gamma_a^{NS}} \left\lceil \frac{t-s_j}{T_j} \right\rceil C_{j,a}$$
4. the sum of the additional computational requirements of all alternate jobs of Γ_a^{NS} executed before time S and which have not finished their executions yet at time S:
$$\sum_{\tau_j \in \Gamma_a^{NS}} max\left[0,\ (R_i(S+s_j-T_j)+s_j-T_j)\right];$$
5. sporadic tasks with higher priorities than τ_i released at time S:
$$\sum_{\tau_j \in hp^{NS}(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j.$$

□

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 (τ_1,τ_2), $g_{1,2}=6$ and condition (1) is satisfied:

$$4 \leq (4-0)mod6 \leq 6-2 \Longrightarrow 4 \leq 5 \leq 5.$$

Thus, Γ^S is schedulable.
According to theorem 2, the schedule of Γ^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 τ_1 in the permanent phase are $\Psi_1' = \{0,12,24\}$, thus $\Psi_1 = \{4,16,28\}$.
The release times of τ_2 in the permanent phase are $\Psi_2' = \{4,22\}$ thus $\Psi_2 = \{6,24\}$.
The critical instants of Γ^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:

$$W_{1,a}(t) = 4 + \left\lceil \frac{t-s_1}{T_1} \right\rceil C_1 + \left\lceil \frac{t-s_2}{T_2} \right\rceil C_2$$

$$= 4 + \left\lceil \frac{t - s_1}{12} \right\rceil 4 + \left\lceil \frac{t - s_2}{18} \right\rceil 2.$$

The response times $R_{1,a}$ of $\tau_{1,a}$ are given by:

S	s_1	s_2	$R_{1,a}(S)$
4	8	0	**6**
16	8	6	4
28	8	12	4

As $R_{1,a} \leq (D_{1,a} = 6)$, $\tau_{1,a}$ is schedulable.

For $\tau_{2,a}$ we have:

$$W_{2,a}(t) = 2 + \left\lceil \frac{t - s_1}{T_1} \right\rceil C_1 + \left\lceil \frac{t - s_2}{T_2} \right\rceil C_2 + \left\lceil \frac{t - s_{1,a}}{T_{1,a}} \right\rceil C_{1,a}$$

$$+ max\left[0,\ (R_{1,a}(S + s_{1,a} - T_{1,a}) + s_{1,a} - T_{1,a})\right]$$

$$= 2 + \left\lceil \frac{t - s_1}{12} \right\rceil 4 + \left\lceil \frac{t - s_2}{18} \right\rceil 2 + \left\lceil \frac{t - s_{1,a}}{12} \right\rceil 4 + \alpha$$

with $\alpha = max\left[0,\ (R_{1,a}(S + s_{1,a} - 12) + s_{1,a} - 12)\right]$.

After calculating $R_{1,a}$ for $S = 6$ and $S = 24$ we obtained:

S	$(S + s_{1,a} - 12)$	$R_{1,a}$	$(s_{1,a} - 12)$	α
6	4	6	-2	4
24	16	4	-8	0

The response times $R_{2,a}$ of $\tau_{2,a}$ are given by:

S	s_1	s_2	$s_{1,a}$	$R_{2,a}(S)$
6	6	16	10	6
24	0	16	4	**10**

As $R_{2,a} \leq (D_{2,a} = 10)$, $\tau_{2,a}$ is schedulable.

For τ_3 we have:

$$W_3(t) = 2 + \left\lceil \frac{t - s_1}{T_1} \right\rceil C_1 + \left\lceil \frac{t - s_2}{T_2} \right\rceil C_2$$

$$+ \left\lceil \frac{t - s_{1,a}}{T_{1,a}} \right\rceil C_{1,a} + \left\lceil \frac{t - s_{2,a}}{T_{2,a}} \right\rceil C_{2,a} + \alpha + \beta$$

with $\alpha = max\left[0,\ (R_{1,a}(S + s_{1,a} - 12) + s_{1,a} - 12)\right]$ and $\beta = max\left[0,\ (R_{2,a}(S + s_{2,a} - 18) + s_{2,a} - 18)\right]$.

After calculating $R_{1,a}$ and $R_{2,a}$ for $S = 6$ and $S = 24$ we obtained:

S	$(S + s_{1,a} - 12)$	$R_{1,a}$	$(s_{1,a} - 12)$	α
0	-8	0	-8	0
12	4	6	-8	0
22	16	4	-6	0

and

S	$(S + s_{2,a} - 18)$	$R_{2,a}$	$(s_{2,a} - 18)$	β
0	-12	0	-12	0
12	6	6	-6	0
24	6	6	-16	0

Thus, we have:

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

The response times R_3 of τ_3 are given by:

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	2	0	6	2	**36**

As $R_3 \leq (D_3 = 36)$, τ_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 τ_3 and the second job of $\tau_{2,a}$ meet their respective deadlines.

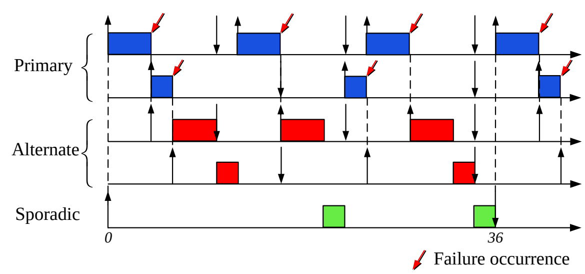

Figure 4. Scheduling diagram within a permanent phase in the worst case faillures occurrences

Figure 5 shows the scheduling diagram of primary, alternate and sporadic tasks within a permanent phase in the case of arbitrary faillures 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.

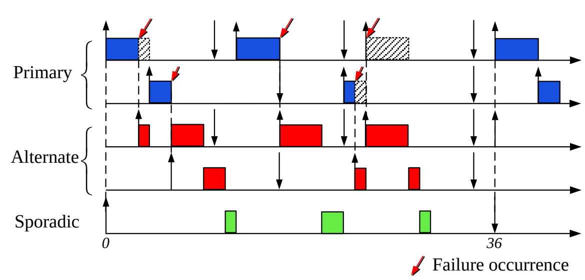

Figure 5. Scheduling diagram within a permanent phase in the case of arbitrary failures occurrences

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 tasks 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

[1] P. Balbastre, I. Ripoll, and A. Crespo. Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. *IEEE Trans. Comput.*, 57:96–109, January 2008.

[2] S. Baruah and S. Chakraborty. Schedulability analysis of non-preemptive recurring real-time tasks. *Parallel and Distributed Processing Symposium, International*, 0:149, 2006.

[3] M. Behnam and D. Isovic. Real-time control design for flexible scheduling using jitter margin, 2007.

[4] A. Burns, R. Davis, and S. Punnekkat. Feasibility analysis of fault-tolerant real-time task sets. In *Real-Time Systems, 1996., Proceedings of the Eighth Euromicro Workshop on*, pages 29 –33, jun 1996.

[5] G. Buttazzo and A. Cervin. Comparative assessment and evaluation of jitter control methods. In *Proc. 15th International Conference on Real-Time and Network Systems*, Nancy, France, Mar. 2007.

[6] F. Eisenbrand, N. Hahnle, M. Niemeier, M. Skutella, J. Verschae, and A. Wiese. The periodic maintenance problem. Technical report, EPF Lausanne and TU Berlin, February 2010.

[7] L. George, N. Rivierre, and M. Spuri. Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling. Research Report RR-2966, INRIA, 1996. Projet REFLECS.

[8] S. Ghosh, R. Melhem, D. Moss, and J. S. Sarma. Fault-tolerant rate-monotonic scheduling. *Real-Time Systems*, 15:149–181, 1998. 10.1023/A:1008046012844.

[9] C.-C. Han, K. Shin, and J. Wu. A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults. *Computers, IEEE Transactions on*, 52(3):362 – 372, march 2003.

[10] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of period and sporadic tasks. In *Proceedings of the 12 th IEEE Symposium on Real-Time Systems*, pages 129–139, December 1991.

[11] M. Joseph and P. K. Pandya. Finding response times in a real-time system. *Comput. J.*, 29(5):390–395, 1986.

[12] J. H. M. Korst, E. H. L. Aarts, and J. K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing*, 8(4):428–435, 1996.

[13] J. H. M. Korst, E. H. L. Aarts, J. K. Lenstra, and J. Wessels. Periodic multiprocessor scheduling. In *PARLE (1)*, pages 166–178, 1991.

[14] B. Lincoln. Jitter compensation in digital control systems. In *American Control Conference, 2002. Proceedings of the 2002*, volume 4, pages 2985 – 2990 vol.4, 2002.

[15] M. Marouf and Y. Sorel. Schedulability conditions for non-preemptive hard real-time tasks with strict period. In *Proceedings of 18th International Conference on Real-Time and Network Systems, RTNS'10*, Toulouse, France, Nov. 2010.

[16] M. Marouf and Y. Sorel. Scheduling non-preemptive hard real-time tasks with strict periods. In *Proceedings of 16th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'2011*, Toulouse, France, Sept. 2011.

[17] P. Marti, J. Fuertes, G. Fohler, and K. Ramamritham. Jitter compensation for real-time control systems. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, pages 39 – 48, dec. 2001.