Firmness Analysis of Real-Time Tasks

AMIR BEHROUZIAN, Eindhoven University of Technology
HADI ALIZADEH ARA, Eindhoven University of Technology
MARC GEILEN, Eindhoven University of Technology
DIP GOSWAMI, Eindhoven University of Technology
TWAN BASTEN, Eindhoven University of Technology & ESI, TNO

(m, k)-firm real-time tasks require meeting the deadline of at least m jobs out of any k consecutive jobs. When compared to hard real-time tasks, (m, k)-firm tasks open up the possibility of tighter resource-dimensioning in implementations. Firmness analysis verifies the satisfaction of (m, k)-firmness conditions. Scheduling policies under which a set of periodic tasks runs on a resource influence the number of deadline missed jobs. Therefore, the nature of the firmness analysis problem depends on scheduling policies. In this work, we present Firmness Analysis (FAn) methods for three common scheduling policies – synchronous and asynchronous Static Priority Preemptive (SPP) policies and Time Division Multiple Access (TDMA). We first introduce the Balloon and Rake problem – the problem of striking the maximum number of balloons in a balloon line with a rake. We show that the common core of firmness analysis problems can be abstracted as the Balloon and Rake problem. Next, we prove that the Finite Point method is a solution to the Balloon and Rake problem. We illustrate how existing FAn methods for the TDMA and asynchronous SPP policies can be adapted to use the same solution framework for the Balloon and Rake problem. Using the solution of the Balloon and Rake problem, we adapt the existing FAn methods to synchronous SPP scheduling policies. The scalability of the FAn methods is compared with that of a timed-automata approach, a brute-force approach and a Mixed Integer Linear Programing method. The FAn methods scale substantially better to firmness analysis problem instances with a large k and a high number of tasks.

Additional Key Words and Phrases: Deadline miss, (m, k)-firm, Balloon and Rake problem, Finite point method, Firmness analysis

ACM Reference Format:
Amir Behrouzian, Hadi Alizadeh Ara, Marc Geilen, Dip Goswami, and Twan Basten. 2020. Firmness Analysis of Real-Time Tasks. 1, 1 (May 2020), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A real-time system provides functional guarantees by executing software on computing resources meeting specified temporal constraints. A real-time system is typically composed of one or more real-time tasks running on a computing resource with predefined deadlines. On a *shared* computing resource, multiple real-time tasks *arbitrate* for resource access and a *scheduler* decides the winner

Authors' addresses: Amir Behrouzian, Eindhoven University of Technology, Eindhoven, The Netherlands, A.R.Baghban.Behrouzian@tue.nl; Hadi Alizadeh Ara, Eindhoven University of Technology, Eindhoven, The Netherlands, s.h.seyyed.alizadeh@tue.nl; Marc Geilen, Eindhoven University of Technology, Eindhoven, The Netherlands, M.C.W.Geilen@tue.nl; Dip Goswami, Eindhoven University of Technology, Eindhoven, The Netherlands, D.Goswami@tue.nl; Twan Basten, Eindhoven University of Technology, & ESI, TNO, Eindhoven, The Netherlands, A.A.Basten@tue.nl.

XXXX-XXXX/2020/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

based on a specific policy. Given a set of real-time tasks and a scheduler, a schedulability analysis deals with the question of meeting temporal constraints in terms of deadlines.

Real-time tasks are categorized into *hard*, *soft* and *firm* in terms of their tolerance to deadline misses [9]. Violation of a single deadline may have catastrophic consequences for hard tasks. The hard schedulability problem has been thoroughly investigated in the last decades [19]. Providing hard guarantees is usually expensive since it requires resource dimensioning considering the worst-case scenario which does not necessarily happen often in many real-time applications. In contrast to hard tasks, soft and firm tasks allow occasional deadline misses. As opposed to soft tasks, the number of misses does not influence the quality of service in firm tasks as long as it is within an acceptable bound. Let a job be an execution instance of a task. As a category of firm tasks, (m, k)-firm tasks [14] must meet the deadline of at least m jobs in any k consecutive jobs. (m, k)-firm tasks allow better-than-worst-case design by tolerating controlled deadline misses.

Firmness analysis verifies the satisfaction of firmness conditions of given (m, k)-firm tasks on executing computing resources. Such an analysis is fundamentally different for various scheduling policies under which a set of tasks is sharing a processor. This difference is due to the fact that the interference of other tasks on the task under firmness analysis is influenced by scheduling policies. For example, [5] addresses firmness analysis of a set of tasks running under TDMA. Periodic tasks running under an SPP policy are divided into synchronous and asynchronous tasks [11]. A set of tasks is said to be *asynchronous* if the relative release times among the tasks are given; the tasks' relative release times are not given for a *synchrounous* task set[1]. [4] proposes the FAn method for firmness analysis of a set of asynchronous tasks running under an SPP policy.

This work builds on and extends [5] and [4]. We first introduce the Balloon and Rake problem which considers a line of non-overlapping balloons of different sizes and a rake with a finite number of equidistant blades. The problem is to find the maximum number of struck balloons with one strike of the rake. We show that the Balloon and Rake problem abstracts a common core challenge in firmness analysis of a set of periodic tasks running under various scheduling policies. Fundamentally, the firmness analysis depends on the alignment between the release of periodic tasks and *availability* of computing resources for their execution. The challenge is to find the alignment which results in the maximum/minimum number of deadline misses. Our analysis finds the worst- or best-case alignment with respect to deadline misses. The Finite Point (FP) method [5] shows that it is sufficient to verify a set of finite alignments to obtain the worst and best cases. The FP method is used in the FAn methods for the TDMA [5] and the asynchronous SPP policies [4] to perform firmness analysis. In this work, using the same principle, we use the Finite Point method to solve the Balloon and Rake problem. We show that the firmness analysis problems for the TMDA and the asynchronous SPP policies can be reduced to the Balloon and Rake problem. The common solution framework of the Balloon and Rake problem is used for the firmness analysis for both policies.

Next, we use the same analysis framework to develop a novel firmness analysis for a task set under the synchronous SPP policy. In this case, we obtain the minimum number of deadline hit jobs in any k consecutive jobs of the task under firmness analysis. Having this number at least equal to m (in an (m, k)-firmness condition) verifies the satisfaction of the firmness condition. In this problem, the main challenge is to obtain the relative release times of all the tasks, which leads to the maximum number of deadline misses in a window of k consecutive jobs of the task under firmness analysis.

[1]We follow the definition in [11] which is opposite to the ones reported in [7][6]. The naming considers the fact that a set of synchronous tasks can be released simultaneously as opposed to asynchronous tasks.

The scalability of our FAn methods is compared with that of three existing approaches: 1) a brute force approach which verifies all the possible alignments between requested and available computing resources in a window of k consecutive jobs inspired by [7]; 2) a timed-automata model of the problem inspired by [4]; 3) a Mixed Integer Linear Programing (MILP)-based method inspired by [23] (for the synchrnous SPP policy). The FAn methods scale substantially better to problem instances with a large k and a high number of tasks than the timed-automata, brute force and the MILP approaches.

The organization of the paper is as follows. Section 2 discusses related work. Section 3 introduces the Balloon and Rake problem and presents the Finite Point (FP) method as a solution to this problem. Section 4 introduces setup under consideration and a number of general definitions relevant for Firmness analysis of TDMA and synchronous and asynchronous SPP policies. Section 5, 6 and 7 present how the firmness analysis problems for TDMA and asynchronous and synchronous SPP policies can be transformed into an instance of Balloon and Rake problem and the corresponding adaptation of the FAn methods, resulting in three different FAn methods. Section 8 presents the experimental results and comparison with the state-of-the-art methods. Section 9 concludes.

2 RELATED WORK

Traditional schedulability analysis methods aim to address the question whether there is a possible deadline miss for a given set of *hard* real-time tasks [19]. These analyses rely on the existence of a *critical instant* – a scenario where all the tasks are activated simultaneously. The critical instant is proven to be the worst-case scenario and it suffices to analyze the particular task activation pattern at the critical instant. That is, if a task set is schedulable (or does not miss *any* deadline) in the critical instant, it is schedulable in all other scenarios. While the hard tasks are relevant for safety-critical applications where a single deadline violation may be catastrophic, there are many applications that tolerate *occasional* deadline misses and treating them as the hard tasks may result in resource over-dimensioning [13]. For example, control applications are commonly modeled as *firm* tasks [8, 21, 25]. Firm tasks allow for deadline misses as long as they do not occur too frequently. (m, k)-firm tasks fall under this category where the frequency of allowed deadline misses is defined in terms of m and k. Hard tasks are a special case of firm tasks for which $m = k$. (m, k)-firm conditions were introduced in the context of scheduling messages in a network by Hamdaoui [14]. Bernat [6] extended such conditions to weakly-hard real-time tasks by introducing other types of constraints in the distribution of misses and hits.

Existing studies related to (m, k)-firm tasks can be divided in two main categories. The first line of work focuses on obtaining m and k parameters based on the robustness of a (control) system [13, 20, 25]. The main challenge addressed in these works is how to guarantee stability and control performance with an (m, k)-firm pattern of deadline misses [12, 21]. The second category verifies the satisfaction of given firmness conditions under a scheduling policy and performs firmness analysis. The firmness analysis depends on the task model and the scheduling policy under consideration. Our work falls into the second category.

The identification of the worst-case scenario is the main challenge in any firmness analysis. The worst-case scenario depends on the specific scheduling policies and the task models. In the above context, a line of work considers periodic task models with a given release time of the first jobs under a static priority preemptive policy or an asynchronous SPP policy [7]. Given the first release time of all tasks, the response times of all the jobs of the task under firmness analysis are obtained. Comparing the number of deadline hits in all the windows of k consecutive jobs, the worst-case is obtained. It suffices to simulate the worst-case execution of the tasks in a bounded time interval to check for deadline misses under an asynchronous SPP policy [18]. Kong [17] performs firmness analysis under a hierarchical dynamic priority preemptive scheduling policy.

Another group of work provides a lower bound on the number of deadline hits using typical worst-case analysis on the execution traces of a set of tasks running under SPP [15, 24, 26] and Static Priority Non-Preemptive [22, 26] policies. Using the execution traces, the system model is obtained as the superposition of a typical periodic behavior, and a sporadic overload. Hard real-time analysis is performed for the typical behavior while firmness analysis is performed when the system is overloaded. Typically, the introduction of an overload model makes the task model artificial in nature and hard to apply in real-life scenarios. All the work above assumes that the relative release times of all tasks including the task under firmness analysis are given.

The firmness analysis of an (m, k)-firm task with unknown relative release time with respect to other tasks under a TDMA policy is addressed in [5]. First, the service curve of the TDMA schedule for a single job of the task is obtained. Based on the FP method, the worst-case release of k consecutive jobs in terms of having the minimum number of deadline hits is then calculated. The FAn method [4] determines the first release time of an (m, k)-firm task under an SPP policy. The first release times of all tasks with higher priorities than that of the task under the firmness analysis are assumed to be given. In the problems addressed in these works, only one unknown relative release time is considered. As opposed to this, the problem of finding release times of all the tasks is solved by a MILP formulation in [23] – we compare our results with this approach. These state-of-the-art methods are tailored for specific policies. We propose a common abstract analysis framework that can be adapted to perform firmness analysis under different policies. Using this framework, we develop a firmness analysis for an asynchronous SPP policy, i.e., the case where relative release times of all tasks are unknown – as considered in [23]. We illustrate the wider applicability of the proposed framework considering the problems addressed in [5] (TDMA policy) and [4] (the synchronous SPP policy).

3 BALLOON AND RAKE PROBLEM

This section introduces the Balloon and Rake problem. We propose a solution to this problem which is based on the FP method [5]. The results of this section are later used in Sections 5, 6 and 7.

3.1 Problem Formulation

Definition 1. *(Balloon and Rake Problem)*

Let a line of infinitely many balloons be formed of N possibly different balloons which periodically repeat infinitely many times with period p. Each balloon is specified by its width and its position in a period of the balloon line. Determine the maximum possible number of balloons that can be struck with one hit by a rake with r identical blades on the line of balloons. The blades of the rake have width 0 and distance d from each other. The number of struck balloons is the number of the blades located on balloons.

For every balloon, we refer to its edge with the lower value on the x axis, as the *left end* and with the higher value of x as the *right end* (see Fig. 1.a). Starting the counting from a random balloon, we denote the i^{th} balloon in a period of the line of balloons with $b_i = (w_i, x_i^L)$ where w_i and x_i^L are its width and left-end position, respectively. Here $i \in [1, N]$. We denote the right end of the balloon by x_i^R assuming $x_i^R \leq x_{i+1}^L$ (i.e., balloons do not overlap) for $i \in [1, N]$. The blades are separated by an equal distance d, which is not restricted by the parameters of the line of balloons such as period p or number of balloons N.

Example 3.1. *Consider a line of balloons formed of two balloons, i.e. $N = 2$; in each period (see Fig.1.a), $b_1 = (2.5, 0)$ and $b_2 = (2, 3)$. This pattern of balloons repeats with period $p = 6$. We intend to obtain the maximum number of struck balloons with one hit by a rake with $r = 5$ blades which have a distance d of 2.5 from each other. Without loss of generality, we assume that a period of the balloon*

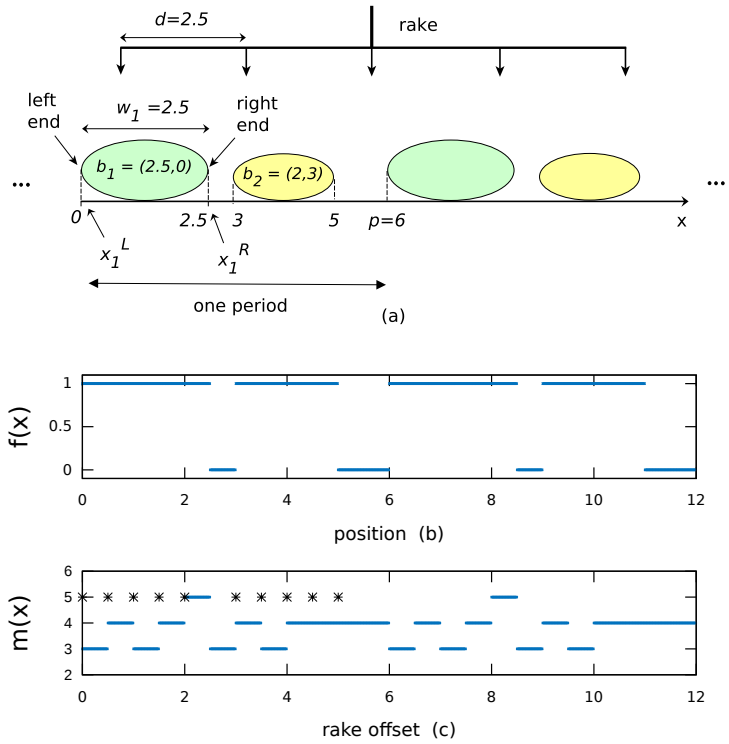

Fig. 1. Example. 3.1. (a) the balloon line and the rake. (b) the balloon function $f(x)$. (c) the strike function $m(x)$. $O_{cnd-max}$ is shown by stars.

line starts at $x = 0$ *and the left end of the balloon* b_1 *is located at* $x_1^L = 0$. *Therefore, there are balloons identical to* b_1 *with left end at* $\ldots, -12, -6, 0, 6, 12, \ldots$.

Definition 2 (Balloon Function). *Balloon function* $f : \mathbb{R} \rightarrow \{1, 0\}$ *takes* 1 *if there exists a balloon at* x *and* 0 *otherwise. Formally*

$$f(x) = \begin{cases} 1 & \text{if } \exists i \in [1, N] : \ \exists z \in \mathbb{Z} : \quad x_i^L + zp \leq x < x_i^R + zp \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that the right end of a balloon is not part of the balloon, as opposed to the left end of it. That is, the intervals where $f(x) = 1$ have close left and open right. Therefore, the intervals where $f(x) = 0$ have close left and open right too. This allows us to determine the minimum number of struck balloons, i.e. the minimum number of blades located at a position x with $f(x) = 1$, by obtaining the maximum number of blades located at a position x with $f(x) = 0$ and subtracting the result from r.

Definition 3 (Strike Function). *Strike function* $m : \mathbb{R} \rightarrow \{1, 0\}$ *denotes the number of struck balloons when the first blade of the rake is located at* x. $m(x)$ *is obtained as follows*

$$m(x) = \sum_{n=0}^{r-1} f(nd + x). \tag{2}$$

We refer to the position of the first blade as the *rake offset*. Fig.1.b and Fig.1.c show $f(x)$ and $m(x)$ in Example 3.1.

3.2 Finite Point (FP) Method

We observe that the FP method introduced in [5] can be adapted to solve the Balloon and Rake problem. Since the balloons are positioned periodically with a period of p, it is concluded that $f(x)$ is also periodic with the same period. From this and Eq. 2, it is concluded that $m(x)$ is also periodic with the same period. Therefore, considering all possible rake offsets in one period of balloons, i.e. $0 \leq x < p$, guarantees obtaining all the possible combinations of struck balloons. However, the number of possible rake offsets in one period of the line of balloons is still infinite. We use the result of the following FP theorem to obtain the maximum number of struck balloons by evaluating $m(x)$ for a finite number of different rake offsets.

Theorem 1. *(Finite Point Theorem) Any increase in the value of $m(x)$ occurs on a rake offset where at least one of the blades is on the left end of a balloon. Formally,*

$$m(x-\epsilon) < m(x) \implies \forall \epsilon > 0 : \exists z \in \mathbb{N} : \exists n \in [1, r] : \\ nd + x - \epsilon < zp + x_i^L \leq nd + x$$

Proof. From $m(x-\epsilon) < m(x)$ and Eq. 1, we conclude that

$$\sum_{n=0}^{r-1} f(nd + x - \epsilon) < \sum_{n=0}^{r-1} f(nd + x). \tag{3}$$

Since the summations in the two sides of the Eq. 3 have the same number of terms, there is at least one n for which $f(nd+x-\epsilon) < f(nd+x)$. Moreover, from Def. 2, $f(x)$ takes only 0 and 1. Therefore, $f(nd + x - \epsilon) = 0$ and $f(nd + x) = 1$. From Eq. 1 we conclude that $nd + x - \epsilon < zp + x_i^L \leq nd + x$ for some z and n. □

Theorem 1 implies that, to obtain the maximum number of struck balloons we need to obtain $m(x)$ only for those rake offsets in one period of the balloon line where a blade is located at the left end of a balloon. Comparing the values of $m(x)$ for these finite number of offsets, we obtain the maximum value of $m(x)$.

Let $O_{cnd-max}$ be a set of rake offsets in which a blade is located on the left end of a balloon, i.e. x_i^L. $O_{cnd-max}$ is obtained as follows

$$O_{cnd-max} = \{mod(mod(x_i^L - nd, p) + p, p) \mid i \in [1, N], n \in [0, r-1]\} \tag{4}$$

Here, *mod* is the modulus operator defined as follows

$$mod(x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor. \tag{5}$$

In Eq. 4, i and n can take N and r different values, respectively. Therefore, $O_{cnd-max}$ can have at most $N \times r$ elements. Thus, to obtain the guaranteed maximum number of struck balloons we need to obtain $m(x)$ for only $N \times r$ different rake offsets.

$O_{cnd-max}$ for Example. 3.1 is shown with stars on Fig. 1.c. The fact that there is at least one star at any offset where there is an increase in the value of $m(x)$ validates the FP method. $O_{cnd-max}$ has $2 \times 5 = 10$ elements. Comparing the value of $m(x)$ for all these rake offsets, we conclude that the maximum of 5 balloons are struck with the rake offsets of 2.

Overall, to solve a Balloon and Rake problem the two main steps are obtaining $O_{cnd-max}$ using Eq. 4 and comparing the value of $m(x)$, obtained from Eq. 2, for all these offsets.

4 SETUP UNDER CONSIDERATION

This section provides background on modeling of periodic tasks, which is inspired from [5]. Let $\Pi = \{\tau_i\}_{1 \le i \le M}$ be a set of M periodic tasks sharing a processor. Each task $\tau_i = (C_i, T_i, D_i)$ is specified by its constant execution time C_i, activation period T_i and execution deadline D_i. We discuss in the successive sections that deviation in the execution time of a task has monotonic effect on our results and we may choose the worst-case execution time in our method. We assume that the deadline of a task is at most its activation period, i.e. $D_i \le T_i$. Each instance of a task is referred to as a *job*. The activation time of each job is referred to as the *release time* of the job. The earliest time t when a job of τ_i has been executed for C_i time unit is referred to as the *response time* of the job. The accessibility of τ_i to a processor at time t is modeled with *accessibility function*.

Definition 4 (Accessibility Function). *The accessibility function $l_i : \mathbb{R} \rightarrow \{0, 1\}$ takes 1 if the processor is accessible to τ_i at t and 0 otherwise.*

Execution completion of a job of τ_i released at t before its deadline depends on the overall access time of the task to the processor from t to $t + D_i$ which is captured by *total accessibility function*.

Definition 5 (Total Accessibility Function). *The total accessibility function $g_i : \mathbb{R} \rightarrow [0, D_i]$ indicates the overall access of τ_i to the processor between t and $t + D_i$. $g_i(t)$ is obtained as follows*

$$g_i(t) = \int_t^{t+D_i} l_i(x)dx. \tag{6}$$

Definition 6 (Deadline Hit job (DH)). *A job of τ_i released at t is a DH if $g_i(t) \ge C_i$. A job of τ_i that is not a DH is referred to as a Deadline Miss job (DM).*

We assume that a job that misses its deadline is not executed. Def. 6 implies that we can decide whether a job of τ_i is a DH immediately after its release. This can be used by a run-time scheduler to detect a DM immediately after a job release and to not execute that job at all if this is the case. Therefore, since we assumed above that the deadline of a task is at most its activation period, once a task has a release at t there is no demanded (unfinished) workload from previous jobs of the task at time t independent of whether or not the previously released job was a DM or a DH. This execution strategy is particularly applicable to feedback control applications where execution of outdated samples can cause instability [3]. Moreover, this helps us to effectively use the processor by executing less.

Definition 7 (hit-zone function). *Hit-zone function $h_i : \mathbb{R} \rightarrow \{0, 1\}$ takes 1 if a job of τ_i released at t is a DH and 0 otherwise. Formally,*

$$h_i(t) = \begin{cases} 1 & if \quad C_i \le g_i(t) \\ 0 & if \quad C_i > g_i(t) \end{cases} \tag{7}$$

Any time interval $t \in [t1, t2)$ with $h_i(t) = 1$ is referred to as a *hit-zone*.

5 FAN METHOD FOR TDMA SCHEDULES

In this section, we first summarize for firmness analysis problem under TDMA policy addressed in [5]. We further show how the solution to the Balloon and Rake problem (described in Section 3) can be used to perform firmness analysis.

5.1 Problem Definition

We consider a TDMA schedule composed of a periodic TDMA wheel with a finite number of time slots. Each task may be allocated one or more slots and may be executed in the allocated slots.

Definition 8 (TDMA Schedule). *A TDMA schedule is specified by a tuple $TDMA = (w, A)$ where w is the length of the periodic TDMA time wheel and A is a non-empty subset of the finite set of intervals $\{(t^i_{start}, t^i_{end}) \in \mathbb{R} \times \mathbb{R} \mid (0 \leq t^i_{start} < t^i_{end} < w) \wedge (1 \leq i \leq T)\}$ such that no two intervals overlap. Each interval is referred to as a time slot and T is the number of time slots in w.*

In Def. 8, we may consider switching overhead [1] between two consecutive time slots. In such a case, $t^i_{end} \neq t^{i+1}_{start}$. Moreover, Def. 8 allows for slots with unequal length which includes a special case with equal slots.

As a timing-compositional scheduling policy, TDMA allows us to analyze each task separately. We consider that the relative release time between every task and the TDMA time wheel is not given. Therefore, a firmness analysis should obtain the minimum possible number of DHs in any k consecutive jobs of an (m, k)-firm task. Having this number equal to or more than m guarantees the satisfaction of the (m, k)-firmness condition.

5.2 Hit-Zone Calculation

Let $\tau_1 \in \Pi$ be a periodic task running under a TDMA schedule $TDMA = (w, A)$ where the set of time slots $\{(t^i_{start}, t^i_{end}), 1 \leq i \leq T\} \subset A$ are assigned to τ_1. From Def. 4 the accessibility function $l_1(t)$ is calculated as follows

$$l_1(t) = \begin{cases} 1 & \text{if } \exists n, i \in \mathbb{Z}: \quad t^i_{start} + nw \leq t < t^i_{end} + nw, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

$g_1(t)$ and $h_1(t)$ are obtained using Eq. 6 and Eq. 7, respectively. From Eq. 8, we conclude that $l_1(t)$ is periodic with the period of w. Therefore, from Eq. 6 and Eq. 7, $g_1(t)$ and $h_1(t)$ are also periodic with the same period.

Example 5.1. *Let $\tau_1 = (2.2, 7, 7)$ be a $(8, 10)$-firm task running under a TDMA schedule $TDMA = (5.5, A)$ where $A = \{(0, 1), (1.1, 2.1), (2.2, 3.2), (3.3, 4.3), (4.4, 5.4)\}$ (times in ms). Note that, there is a switching overhead of $0.10ms$ between every two consecutive slots. The second and the forth time slots are allocated to τ_1. Fig. 2.a depicts the accessibility function $l_1(t)$ obtained from Eq. 8. Substituting $l_1(t)$ in Eq. 6 results in $g_1(t)$ in Fig. 2.b. Fig. 2.c shows $h_1(t)$.*

5.3 Obtain The Minimum Number of DHs

In order to obtain the minimum number of DHs in any k consecutive jobs of τ_1 we must obtain the minimum number of jobs that are released in hit-zones out of any k consecutive jobs. We refer to the Balloon and Rake problem discussed in Section 3. In order to connect the problem above to the Balloon and Rake problem we should define which parameters represent rake blades and balloon function f. In that context, each release of τ_1 is considered as a blade of a rake with k blades for k consecutive jobs. The distance between every two blades is T_1 (i.e. period of τ_1). Next, we obtain balloon function. Recall that Section 3 obtains maximum number of struck balloons. However, in this section we intend to obtain the minimum number of the jobs of τ_1 released at hit zone, i.e. the intervals where $h_1(t) = 1$. In this regard, we first obtain the maximum number of jobs of τ_1 released at the intervals where $h_1(t) = 0$. In other words we assume that the intervals where $h_1(t) = 0$ are balloons, i.e. $f = 1 - h_1$. Then we obtain the maximum number of struck balloons and subtract the result from k. This provides the minimum number of the jobs of τ_1 released at hit zones in any k

Table 1. Balloon and Rake problem parameters corresponding to the FAn method for TDMA schedule

type	Balloon and Rake	FAn: TDMA schedule
input	N and p in the form of f	$1 - h_1$
	# of rake blades, r	k
	distance between blades, d	T_1
output	max # of struck balloons	k − min # of DHs

consecutive jobs. Table 1 lists the parameters of the Balloon and Rake problem corresponding to those used in this section.

Fig. 2.d shows the number of DHs against the offset of a window of 10 consecutive jobs of τ_1. Moreover, the stars on Fig. 2.d shows the candidate offsets of the minimum number of DHs obtained from the FP method. The fact that there is a star at any point where the number of DHs decreases validates the FAn method.

We obtain the minimum possible number of DHs with an assumption that τ_1 has a constant execution time. Variation in the execution time of τ_1 has a monotonic effect on the number of DHs [5]. Therefore, we may choose the worst-case execution time in the FAn method which still results in the minimum number of DHs. That is, in a case of a change in either of the relative release time and execution time of τ_1, the number of DHs may increase. In such a case, the FAn method therefore yields conservative results which are used to validate the satisfaction of the (m, k)-firm condition. In Example 5.1, the FAn method results in the minimum of 7 DHs for τ_1 in any 10 consecutive jobs which implies that $(8, 10)$-firmness condition of τ_1 is not satisfied.

6 FAN METHOD FOR ASYNCHRONOUS TASKS

This section illustrates the extended FAn method for firmness analysis of asynchronous tasks under an SPP policy addressed in [4].

6.1 Problem Definition

This section proposes FAn method for the firmness analysis of an (m, k)-firmness task $\tau_i = (C_i, T_i, D_i, O_i)$ in a set of *asynchronous* tasks $\Pi = \{\tau_i\}_{1 \le i \le M}$ running under an *SPP scheduling policy*. Note that each asynchronous task is specified with one more parameter O_i compared to the task model introduced in Section 4. O_i indicates the first release time of the task τ_i relative to the first release time of the highest priority task.

Definition 9 (SPP Schedule). *An SPP schedule for M periodic tasks is defined as the set $SPP = \{(\tau_i, P_i)\}_{1 \le i \le M}$ of M tuples. Each tuple (τ_i, P_i) assigns the priority P_i to the task τ_i. A job of τ_i released at t' is executed at any time t (where $t' \le t < t' + D_i$) if: A) $h_i(t) = 1$; B) τ_i has been executed no more than C_i time unit before t; C) there is no demanded workload from a task with a higher priority than that of τ_i.*

Adding a given (m, k)-firm task τ_1 to a set of asynchronous tasks running under an SPP scheduling policy involves deciding relative first release time of τ_1. The first release time of τ_1 influences the number of DHs as it affects the interference of the tasks with higher priorities than τ_1. In this section, we obtain the first release time O_1 which results in the maximum number of DHs in k consecutive jobs of τ_1.

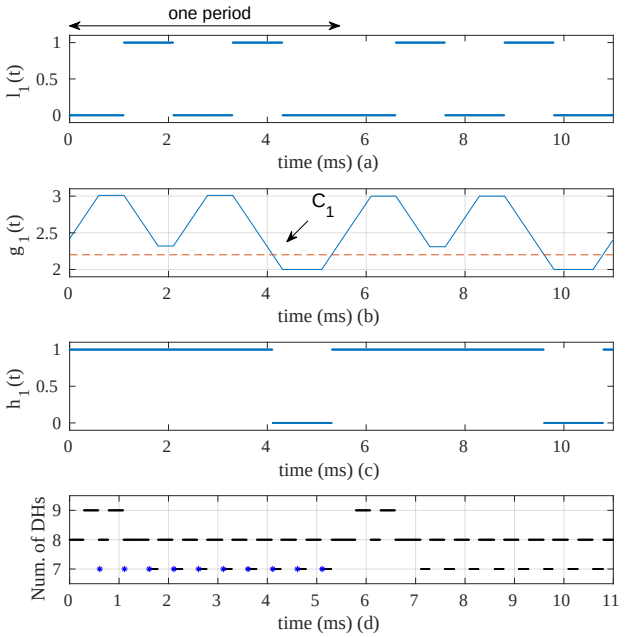

Fig. 2. Example 5.1. (a) accessibility function $l_1(t)$. (b) total accessibility function $g_1(t)$. (c) hit-zone function $h_1(t)$. (d) number of DHs against the offset of 10 consecutive jobs. The stars show condidate offsets for minimum number of DHs.

6.2 Hit-Zone Calculation

The relative release times of all the tasks with higher priorities than τ_1 are specified by the parameters O_i and are given. Therefore, we can obtain their interference on a job of τ_1 released at t. This interference is periodic with the period of the hyper-period of all the tasks with higher priorities than τ_1 which is specified with H_{1^+}. Formally, $H_{1^+} = lcm\{T_j | j \in hp(1)\}$ where $hp(1)$ is the set of all the priorities higher than 1. Generally, the existence of H_{1^+} is not obvious when the tasks are activated by different possibly asynchronous clocks (e.g., in multiprocessors). We justify the existence of H_{1^+} based on the fact that we deal with uniprocessor scheduling triggered by a single clock where the activation period of every task is an integer multiple of the system clock cycle. Therefore, the accessibility function (see Section 4) can be obtained as follows

$$l_1(t) = \begin{cases} 0 & \text{if } \exists j \in hp(1) : \exists n \in [1, M_{j-hyp}] : \quad B_j^n \le t < R_j^n \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

where B_j^n and R_j^n are the release time and the response time of the n^{th} job of the tasks τ_j with $j \in hp(1)$. Here $M_{j-hyp} = H_{1^+} \,/\, T_j$ is the number of releases of τ_j within $[0, H_{1^+})$. The release time of each job is $B_j^n = nT_j + O_j$. [7] proposes a method to obtain R_j^n for each job. While efficient for the individual jobs, this method is not efficient when we obtain R_j^n for all the jobs of the tasks τ_j with $j \in hp(1)$ within $[0, H_{1^+})$ since a part of calculations is repeated for every job in a set of consecutive activations. We propose an iterative method instead where calculation of R_j^n and $l_1(t)$ are interdependent as explained in the following.

We initially assume that $l_1(t) = 1$ for $t \in [0, H_{1^+})$. Starting from the first job of the highest priority task released at $t = 0$, in each iteration we first obtain the response time of a job. Second, we update $l_1(t)$ by considering $l_1(t)|_{B_j^n \le t < R_j^n} = 0$. Third, we use the updated $l_1(t)$ to obtain the response time of the next job of the same task in one hyper-period H_{1^+}. This process is elaborated as follows.

Let τ_{hst} be the highest priority task. Without loss of generality, we assume that the first release time of t_{hst} is 0, i.e. $B_{hst}^1 = 0$. $R_{hst}^1 = C_{hst}$ as τ_{hst} is not preempted by any other task. Therefore, $l_1(t)|_{0 \le t < C_{hst}} = 0$. R_j^n for the second job of τ_{hst} and all jobs of tasks τ_j with $j \in hp(1)$ in the first hyper-period H_{1^+} equals to the smallest $t > 0$ satisfying the following equation [7]:

$$t = nC_j + Idle_j(B_j^n) + \sum_{hj \in hp(j)} \left\lceil \frac{t - O_{hj}}{T_{hj}} \right\rceil C_{hj}. \tag{10}$$

The three terms in the right hand side of Eq. 10 are explained as follows. nC_j is the execution time of the n jobs of τ_j. $Idle_j(B_j^n)$ is the overall time when the resource is not used to execute any job of τ_j or $hp(\tau_j)$ in $[0, B_j^n]$ which is obtained as follows.

$$Idle_j(B_j^n) = \int_0^{B_j^n} l_1(t)dt \tag{11}$$

The last part of Eq. 10 models interference due to the tasks with higher priority than τ_j and hj is the element of $hp(j)$, i.e., the set of all priorities higher than j. The summation in Eq. 10 captures the overall interference of the jobs of the tasks with a higher priority than τ_j within $[0, t]$. Eq. 10 can be solved using the fixed point iteration method as follows.

$$\begin{cases} t^{(0)} = nC_j + Idle_j(B_j^n) \\ t^{(\gamma+1)} = nC_j + Idle_j(B_j^n) + \sum\limits_{hj \in hp(j)} \left\lceil \dfrac{t^{(\gamma)} - O_{hj}}{T_{hj}} \right\rceil C_{hj} \end{cases} \tag{12}$$

where $\gamma \in \mathbb{Z}^+$. We calculate a new value of t^γ in each iteration by incrementally increasing γ until $t^{\gamma+1} = t^\gamma$ is satisfied[2].

Once we obtained R_j^n, we update $l_1(t)$ and continue with obtaining the response time of R_j^{n+1}. After obtaining R_j^n for all jobs of τ_j in one hyper-period H_{1^+} and updating $l_1(t)$ correspondingly, we obtain the response time of the first job of the next highest priority task τ_j with $j \in hp(1)$. We continue this process until we obtain the response time of all the jobs of the tasks with higher priorities than τ_1 in one hyper period and update $l_1(t)$ for all of these jobs. The final result of $l_1(t)$ is used as the accessibility function of τ_1.

Let us consider the case where there is an (m, k)-firm task τ_j with higher priorities than τ_1 in the set Π. Recall that DMs are never executed. We need to take this assumption into account in the process of obtaining the response time of the tasks τ_j with $j \in hp(1)$ and updating $l_1(t)$ that mentioned previously in this section. In this regard, for DMs of τ_j we assume that $B_j^n = R_j^n$.

Example 6.1. *We intend to add $\tau_1 = (1.5, 13, 11, O_1)$ to a set of asynchronous tasks $\Pi = \{\tau_2, \tau_3\}$ where $\tau_2 = (3, 8, 7, 4)(\times 100\mu s)$ and $\tau_3 = (2, 6, 3, 0)(\times 100\mu s)$ are running under an SPP policy with priorities $P_3 > P_2 > P_1$. We intend to obtain an offset O_1 where τ_1 has the maximum possible number of DHs in any k consecutive jobs. Fig. 3.a depicts the execution of τ_2 and τ_3 in one hyper period of $H_{1^+} = 24$. $Idle_2(B_2^n)$ and $Idle_3(B_3^n)$ are denoted by Id_2 and Id_3 on the figure, respectively. Fig. 3.b shows the accessibility function $l_1(t)$.*

[2]Note that, no switching overhead is considered in this solution.

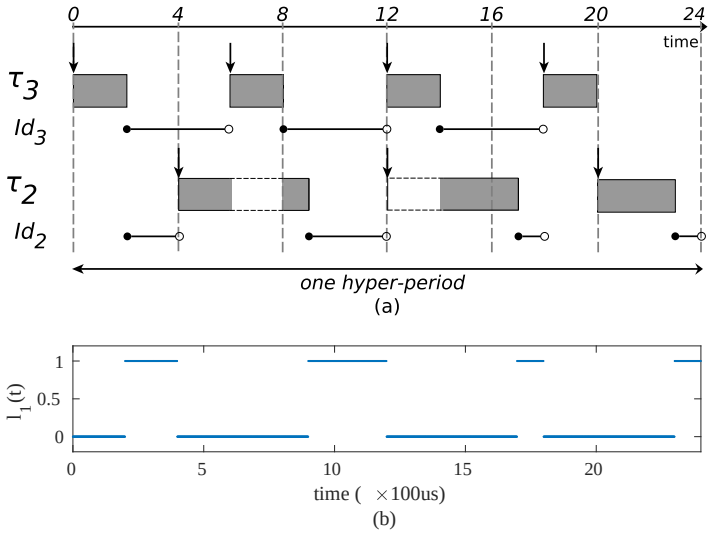

Fig. 3. Calculation of $l_1(t)$ in Example 6.1. (a) Id_2 and Id_3 show the intervals where $Idle_2 = 1$ and $Idle_3 = 1$, respectively. (b) $l_1(t)$ in one hyper-period.

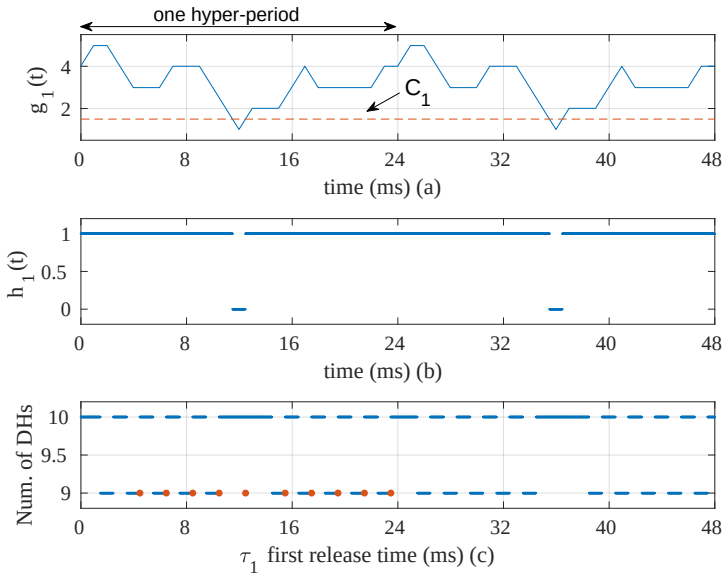

Fig. 4. Results of the FAn method applied on Example 6.1. $g_1(t)$ (a), $h_1(t)$ (b), and the number of DHs (c) in two hyper-period of $H_{1^+} = 24$.

Once the accessibility function is obtained, the total accessibility function and hit zone function can be obtained using Eq. 6 and Eq. 7. Fig. 4.a and Fig. 4.b show $g_1(t)$ and $h_1(t)$ for Example 6.1, respectively.

Table 2. Balloon and Rake problem parameters corresponding to FAn method for multiple asynchronous tasks

type	Balloon and Rake	FAn: asynchronous tasks
input	N and p in the form of f	h_1
	# of rake blades, r	k
	distance between blades, d	T_1
output	max # of struck balloons	max # of DHs
	offset of the rake	first release time of τ_1

6.3 Calculation of The Maximum Number of DHs

For an (m, k)-firm task that is intended to be added to a set of asynchronous tasks, we need to obtain the maximum number of DHs and the corresponding first release time. Having the maximum number of DHs bigger than or equal to m verifies the (m, k)-firm condition. In order to obtain the maximum number of DHs in any k consecutive jobs of τ_1 we must obtain the maximum number of jobs which are released in hit-zones in any k consecutive jobs. This problem can be solved using the FP method (see Section 3) and substituting the parameters of this section corresponding to those of Balloon and Rake problem as listed in Table 2. In this regard, we consider a window of k consecutive jobs of τ_1 as a rake with k blades. Therefore, the distance between every two consecutive blades equals to T_1. Moreover, we consider each hit zone as a balloon. Therefore, the hit zone function h_1 represents the balloon function f. Fig. 4.c shows the number of DHs in 10 consecutive jobs of τ_1 against the first release times of τ_1 in two hyper-period $2H_{1^+}$. The dots on Fig. 4.c show the candidate offsets for the maximum number of DHs in 10 consecutive jobs of τ_1 and the corresponding number of DHs. The fact that there is at least one dot at any time where there is an increase in the number of DHs, validates the results of FAn method.

In [4], it is proven that in a given set of asynchronous tasks variation in the execution time of any task has monotonic effect in the number of DHs. Considering the worst-case execution time for all tasks, therefore results in the minimum-maximum number of DHs. That is, the number of DHs may increase with a lower execution time of any task with the same or the higher priority than τ_1. Thus, the obtained results from the FAn method is a lower bound for the maximum number of DHs.

7 FAN METHOD FOR SYNCHRONOUS TASKS

This section uses the solution of the Balloon and Rake problem to address the firmness analysis of multiple synchronous tasks running on an SPP policy.

7.1 Problem Definition

This section proposes the FAn method for a set $\Pi = \{\tau_i\}_{1 \le i \le M}$ of multiple *synchronous* tasks $\tau_i = (C_i, T_i, D_i)$ running under an SPP schedule (see Def. 9). Note that, in contrast to asynchronous tasks, relative first release times are not given for synchronous tasks. Let $\tau_1 \in \Pi$ be the lowest priority task which is under firmness analysis. The highest priority task is denoted by τ_{hst}. The set of all the other tasks with priorities higher than τ_1 and lower than τ_{hst} is denoted by Π_{mdl}. The tasks with higher priorities than τ_1 may or may not have hard deadlines. Formally,

$$\Pi = \{\tau_{hst}, \tau_1\} \cup \Pi_{mdl}, \text{ such that } \{\tau_{hst}, \tau_1\} \cap \Pi_{mdl} = \emptyset.$$

Different relative release time of the tasks with higher priorities than τ_1 leads to different interferences of them on τ_1. Therefore, to guarantee the satisfaction of the firmness conditions,

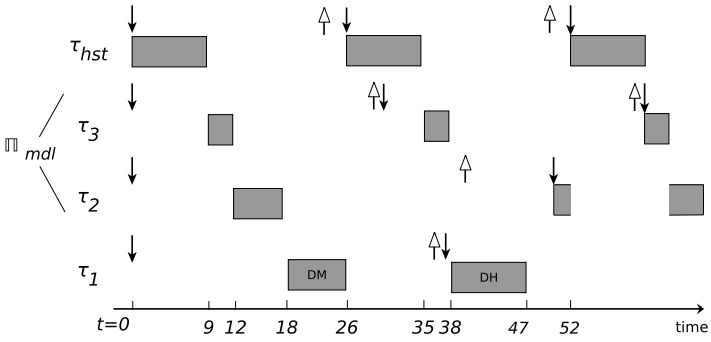

Fig. 5. The critical instant for the tasks in Exampe 7.1. Release times of the tasks and their deadlines are shown with downward and upward arrows, respectively.

we need to identify the worst-case release time that leads to the minimum number of DHs in a window of k consecutive jobs.

The worst-case first release time – also known as *critical instant* [11] – for a single job of the task τ_1 in a set of synchronous tasks occurs when τ_1 and all the tasks with higher priorities than τ_1 have simultaneous releases. This job then experiences the Worst Case Response Time (WCRT) that is denoted by $WCRT_1$ and equals to the minimum $t > 0$ that satisfies the following equation

$$t \geq C_1 + \sum_{\tau_j \in hp(\tau_1)} \left\lceil \frac{t}{T_j} \right\rceil C_j. \tag{13}$$

The fixed point iteration method is used to solve this equation as for Eq.12. $WCRT_1 > D_1$ implies that τ_1 has a DM in the critical instant. The results of the above WCRT analysis would be an initial step to detect whether it is possible for τ_1 to have a DM. Critical instant, however, has no correlation with the minimum number of DHs in k consecutive jobs of τ_1.

Example 7.1. *We take* 4 *synchronous tasks* $\tau_1 = (9, 38, 37), \tau_2 = (6, 50, 40), \tau_3 = (3, 31, 30)$ *and* $\tau_{hst} = (9, 26, 23)$ *where* $P_{hst} > P_3 > P_2 > P_1$ *(all times are in ms).* τ_1 *is an* $(7, 10)$*-firm task. In this example,* $\Pi_{mdl} = \{\tau_2, \tau_3\}$*. We intend to obtain the minimum number of DHs in any* 10 *consecutive releases of* τ_1*. Without loss of generality, we assume that* τ_{hst} *has a release at* $t = 0$*. Fig. 5 depicts the critical instances for all the tasks.* τ_1 *might experience DMs, as shown on the figure.*

In the rest of this section, we first obtain the sufficient conditions for having a DH job without having to consider the relative release time of all the tasks with higher priorities than τ_1. We then use this method to identify a set of periodic intervals in which a release of τ_1 is definitely a DH. Finally, we formulate the problem in the form of a Balloon and Rake problem and obtain a lower bound for the minimum number of DHs in any k consecutive jobs of τ_1.

7.2 Interference Model

For a job of τ_1 released at t, we consider a time window of the length D_1 – starting from t – when the job can be executed before its deadline. Therefore, the job is a DH if

$$D_1 \geq C_1 + intf_{hst}(D_1, t) + \sum_{\tau_i \in \Pi_{mdl}} intf_{ub-\tau_i}(D_1), \tag{14}$$

where $intf_{hst}(D_1, t)$ is the overall interference of τ_{hst} within $[t, D_1)$. $intf_{ub-\tau_i}(D_1)$ is defined as follows.

Definition 10 (Upper bound Interference (UI)). *Let τ_i be a periodic task in a set of synchronous tasks running under an SPP policy. UI function $intf_{ub-\tau_i} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ specifies an upper bound of the overall time that τ_i is executed in a time interval of the length δ.*

$intf_{ub-\tau_i}(D_1)$ is an upper bound of the interference of $\tau_i \in \Pi_{mdl}$ in any window of the length D_1 which is independent of t. Although it is a pessimistic assumption, considering a constant value for the interference of all $\tau_i \in \Pi_{mdl}$ reduces the complexity of the problem as we ignore the relative release time between each of these tasks and τ_1. Next, we obtain $intf_{hst}(D_1, t)$ and $intf_{ub-\tau_i}(D_1)$, respectively.

$intf_{hst}(D_1, t)$ is obtained as follows

$$intf_{hst}(D_1, t) = \int_t^{t+D_1} \beta(x)dx \tag{15}$$

where the function $\beta : \mathbb{R}^+ \rightarrow \{1, 0\}$ models the execution time of τ_{hst} by taking 1 at any time t when τ_{hst} is executed and 0 otherwise. Since τ_{hst} has the highest priority in Π, it is not preempted by any other task. Therefore, $\beta(t)$ is periodic with the same period as τ_{hst}, i.e. T_{hst}. $\beta(t)$ for one period of τ_{hst} starting from 0, i.e. $0 \leq t < T_{hst}$, is calculated as follows

$$\beta(t) = \begin{cases} 1 & \text{if} \quad 0 \leq t < C_{hst} \\ 0 & \text{if} \quad C_{hst} \leq t < T_{hst} \end{cases} \tag{16}$$

From Eq. 15 and the fact that $\beta(t)$ is periodic, it is concluded that $intf_{hst}(\delta, t)$ is also periodic with the same period, i.e. T_{hst}. Fig. 8.a and Fig. 8.b depicts $\beta(t)$ and $intf_{hst}(D_1, t)$ for two periods of τ_{hst} in Example.7.1.

Next, we obtain $intf_{ub-\tau_i}(D_1)$. We first introduce the notion of *upper bound interference* of a task τ_i on a task with a lower priority than that of τ_i. Considering $\tau_i \in \Pi_{mdl}$, $intf_{ub-\tau_i}(D_1)$ is then equivalent to an upper bound of the interference of τ_i on any job of τ_1. Inspired by the worst-case scenario formulated in multiprocessor scheduling [27], we present the following lemma to be used to calculate $intf_{ub-\tau_i}(\delta)$.

Lemma 1. *Let $\tau_i = (C_i, T_i, D_i)$ be a periodic task in a synchronous task set under an SPP policy with the worst-case response time $WCRT_i \leq T_i$. An upper bound for overall execution of τ_i in an interval of the length δ is obtained by considering the beginning of the interval at $t + WCRT_i - C_i$ when a job of τ_i – released at t – is executed between $t + WCRT_i - C_i$ and $t + WCRT_i$ and all the successive jobs are executed immediately after their releases (see Fig. 6).*

Proof. Let α_i^n denotes the n^{th} job of τ_i released at t (see Fig. 6). $t + WCRT_i$ is then the latest possible response time of α_i^n where $WCRT_i$ denotes the worst-case response time of τ_i. Therefore, the minimum possible time difference between the response of one job and the execution commencement of the next job is $T_i - WCRT_i$. This is the scenario where the next job, i.e. α_i^{n+1}, commences its execution immediately after its release, i.e. at $t + T_i$. Therefore, the minimum interval in which $2C_i$ execution of τ_i occurs equals to $T_i + 2C_i - WCRT_i$ which is the interval starting from $t + WCRT_i - C_i$ as shown in Fig. 6. Then, $intf_{ub-\tau_i}(T_i + 2C_i - WCRT_i) = 2C_i$.

An upper bound for the closest executions to these two executions are the successive releases after α_i^{n+1} if they are completely executed immediately after their releases (e.g. the execution of α_i^{n+2} in Fig. 6). Therefore, an upper bound for the execution of τ_i in an interval of length δ is obtained by considering the scenario when a job of τ_i is executed between $t + WCRT_i - C_i$ and $t + WCRT_i$ while the successive jobs are executed without preemption immediately after their

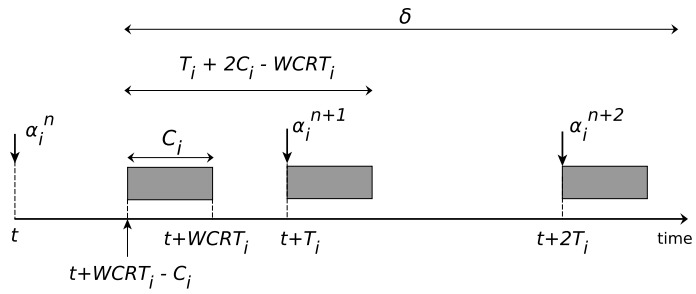

Fig. 6. An execution scenarion for τ_i which results in an upper bound for the execution of the task in an interval with the length δ.

releases. In such a scenario, the interval of the length δ starting from $t + WCRT_i - C_i$ contains an upper bound execution of τ_i. □

$intf_{ub-\tau_i}(\delta)$ is obtained as follows

$$intf_{ub-\tau_i}(\delta) = min\{\delta, C_i\} + max\left\{0\ ,\ \left\lfloor \frac{\gamma}{T_i} \right\rfloor C_i + min\{mod(\gamma, T_i), C_i\}\right\} \tag{17}$$

where

$$\gamma = \delta + WCRT_i - T_i - C_i.$$

Eq. 17 is explained as follows. From Lemma 1, we consider the case where a job of τ_i released at t is executed between $t + WCRT_i - C_i$ and $t + WCRT_i$ (see Fig. 6). The successive jobs of τ_i are executed immediately after their releases. In Eq. 17, the first term, i.e. $min\{\delta, C_i\}$, captures the execution of τ_i within $[t + WCRT_i - C_i, t + T_i)$. $\delta < T_i - (WCRT_i - C_i)$ – which is equivalent to $\gamma < 0$ – is corresponding to the case where the interval with the length δ starting from $t + WCRT_i - C_i$ ends before the release of the next job which occurs at $t + T_i$. In such a case, $intf_{ub-\tau_i}(\delta) = C_i$; unless $\delta < C_i$ which results in $intf_{ub-\tau_i}(\delta) = \delta$. The second term in Eq. 17, with the *max* operator, results in a non-zero value if $\delta \geq T_i - (WCRT_i - C_i)$ (i.e. $\gamma \geq 0$) which is the case when there are more than one execution of τ_i in the interval of length δ. The executions of all the successive jobs are captured with this term. $\lfloor \gamma / T_i \rfloor C_i$ captures the execution of all the jobs released in the interval except the first and the last jobs. Finally, the execution of the last job that is released within the time interval δ is captured with $min\{mod(\gamma, T_i), C_i\}$ in Eq. 17. Note that for any $\tau_i \in \Pi_{mdl}$ the minimum value of $intf_{ub-\tau_i}(D_1)$ equals $min\{D_1, C_i\}$. That is, a job of τ_1 is preempted by at least one complete execution of $\tau_i \in \Pi_{mdl}$ in the worst-case interference, unless $D_1 < C_i$ (see Eq. 17).

In Example 7.1, $WCRT_3 = 12ms$ and $WCRT_2 = 18ms$ which are obtained from Eq. 13. Let both τ_2 and τ_3 have releases at $t = 0$. Considering an interval with the length $D_1 = 37$ starting from $t + WCRT_3 - C_3 = 9$ and $t + WCRT_2 - C_2 = 12$, two executions of τ_3 and one execution of τ_2 occurs at most in the interval as shown in Fig. 7. Therefore,

$$\sum_{\tau_i \in \Pi_{mdl}} intf_{ub-\tau_i}(D_1) = intf_{ub-\tau_2}(D_1) + intf_{ub-\tau_3}(D_1) = C_2 + 2C_3 = 12.$$

Fig. 8.c depicts the two sides of Eq. 14 for Example 7.1. The straight line in the figure shows D_1.

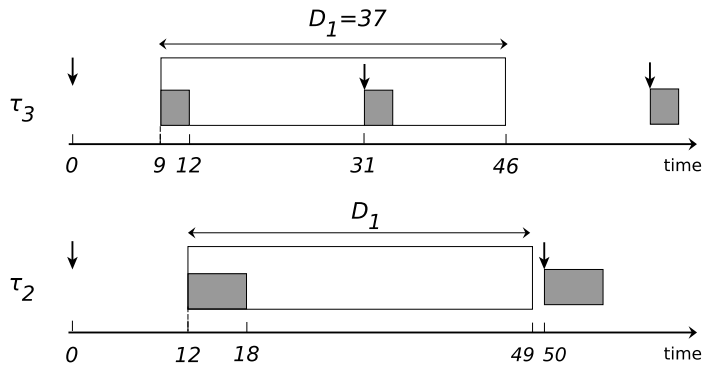

Fig. 7. Calculation of the upper bound interference of τ_2 and τ_3 in any interval of the length D_1.

7.3 Common Hit-Zone Calculation

From Section 4, if a job of τ_1 released at t meets its deadline, t is in a hit-zone. Besides, Eq. 14 provides sufficient condition to decide whether this job is a DH. That is, if the job is not a DH based on Eq 14, it is not necessarily a DM. This case then depends on the releative release time of the tasks in Π_{mdl}. Therefore, using Eq. 14 we obtain a subset of hit zone intervals which we refer to as *common hit-zones*, since they are common among all the hit-zones obtained by considering different release time of the tasks in Π_{mdl}.

Definition 11. *(Common Hit-Zone (CHZ) function) Let τ_1 be in the set Π of multiple synchronous tasks. Common hit-zone function $h'_1(t) : \mathbb{R} \rightarrow \{0, 1\}$ specifies common hit zones by taking $h'_1(t) = 1$ for any time t when Eq. 14 is satisfied.*

Fig. 8.d shows $h'_1(t)$ for Example 7.1 that is obtained by comparing the two sides of Eq. 14 that is shown in Fig. 8.c.

7.4 Calculation of The Minimum Number of DHs

Obtaining the minimum number of jobs in any k consecutive jobs of τ_1 that can be released in CHZ, i.e. when $h'_1 = 1$, provides a conservative bound for the minimum number of DHs. Having the minimum number of DHs more than or equal to m (in (m, k)-firm conditions), guarantees the satisfaction of (m, k)-firm conditions. From the fact that $Int f_{hst}(D_1, t)$ – the only time-dependent term in Eq. 14 – is periodic with the period T_{hst}, we conclude that h'_1 is also periodic with the same period. Then, we refer to the Balloon and Rake problem described in Section 3. That is, we consider a window of k consecutive jobs of τ_1 as a rake with k blades. There is a distance T_1 between every two consecutive blades of the rake. The Balloon and Rake problem obtains the maximum number of struck balloons, however we intend to obtain minimum number of τ_1 releases at CHZs as in Section 5. Therefore, we take $1 - h'_1$ as the balloon function f. Subtracting k from the result of the Balloon and Rake problem with the parameters above therefore results in the minimum number of τ_1 released in CHZs. Table 3 lists the mapping of parameters between the analysis presented in this section and the Balloon and Rake problem.

Fig. 8.e shows the number of DHs in a window of 10 consecutive jobs of τ_1 within two periods of τ_{hst} execution, i.e. $[0, 52)$ in Example 7.1. The stars on the figure show the candidate offsets of the window to obtain the minimum number of jobs released at CHZ. The fact that there is at least one star in any point where there is a decrease in the number of DHs validated the FAn method.

Table 3. The parameters of Balloon and Rake problem corresponding to those of FAn for synchronous tasks

type	Balloon and Rake	FAn: synchronous tasks
input	N and p in the form of f	$1 - h'_1$
	# of rake blades, r	k
	distance between blades, d	T_1
output	max # of struck balloons	$k-$ min # of DHs

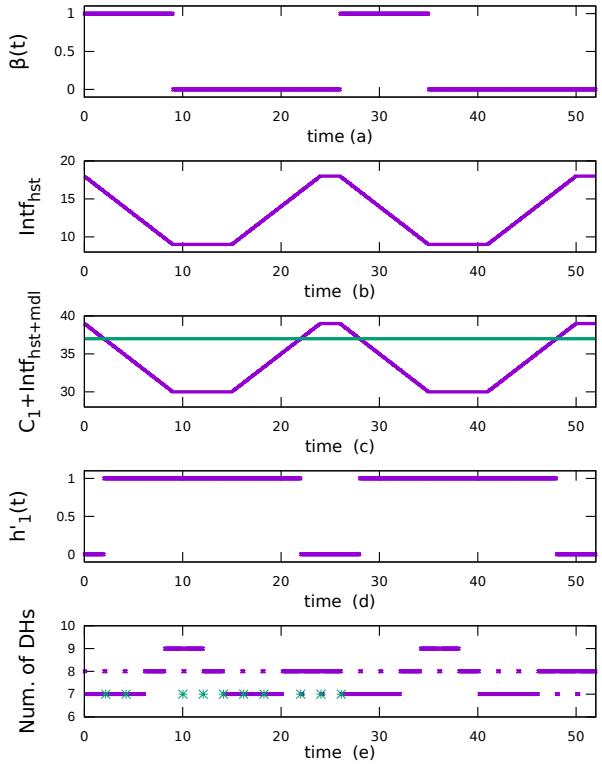

Fig. 8. The solution of Example 7.1 using the FAn method. (a) $\beta(t)$ obtained from Eq. 16. (b) Interference of τ_{hst} on τ_1. (c) Both sides of Eq. 14 for Example 7.1. The straight line indicates D_1. (d) CHZ function. (e) Number of DHs versus the offset of the windows of 10 consecutive jobs of τ_1. Stars show the candidate offsets for the minimum number of DHs.

8 EXPERIMENTAL RESULTS

In this section, we evaluate the scalability and accuracy of the FAn methods. We first explain three existing approaches: (i) a Brute-Force (BF) search approach on all possible relative alignments between tasks inspired by [7], (ii) a method that formulates the firmness analysis problem as a MILP instance, inspired by [23] and, (iii) a reachability analysis based on the Timed-Automata (TA) model of the problem realized in the Uppaal tool inspired by [5]. Second, we provide hypotheses on the complexity of the FAn methods and the existing methods. Third, we introduce our experimental

setups used for evaluation purposes. Finally, we discuss the accuracy and the scalability of all the methods based on our experiments.

8.1 Brute-Force Search Approach

Bernat et al. [7] propose a firmness analysis method for a given alignment between all the tasks in a task set. For a set of tasks running on a shared processor, all the timing parameters, e.g. job releases, execution commencement, execution time, task period, etc., are integer multiples of a processor clock cycle. This allows us to obtain all the possible alignments between τ_1 (i.e. the task under firmness analysis) and the TDMA time wheel, in a TDMA policy, and all the possible alignments between all tasks, in an SPP policy. Then, for each alignment, we use the method in [7] to obtain the minimum (in TDMA and synchronous SPP policies) and the maximum (in the asynchronous SPP policy) number of DHs in any k consecutive jobs of τ_1. Comparing the results of this method for all the possible alignments, we verify the satisfaction of a given (m, k)-firmness condition.

8.2 MILP-Based Method

Sun et al. [23] recently proposed a MILP-based method for the feasibility test of a given (m, k)-firm periodic task in a set of synchronous tasks running under an SPP policy. The main difference between the problem addressed in [23] and that of the FAn method is the task model under consideration. [23] assumes that a job of a task that has missed the deadline is executed completely. However, in our task model, a task is executed only if it does not miss its deadline. We adapted the MILP-based method [23] to the task model considered in this work by adapting the time constraint on the access of a job to the processor. As explained in Section 4 under Def. 6, we assume that a job is not executed if it misses its deadline. The execution of the k^{th} job of task τ_i, J_k, is modeled in the MILP formulation of [23] by ensuring that the task is executed for a total time either equal to C_i before its deadline D_i or 0. This is implemented in the MILP formulation by multiplying C_i with $(1 - b_k)$, where b_k is a boolean value that takes 0 if the task meets its deadline and 1 otherwise. Moreover, we adapted the MILP formulation to provide the minimum number of DHs over all possible synchronizations scenarios. The MILP-based method is based on an over-approximation which results in a conservative output. However, comparison between the result of the MILP method and the BF method shows exact results in the majority of our experiments.

8.3 Timed-Automata Model

The nature of the firmness analysis problems allows us to model the system with timed automata [2]. Our model is inspired by timed-automata model proposed in [5] and [4] which model a set of periodic tasks running under TDMA and SPP policies, respectively. We use the Uppaal tool to verify properties of our model. Run-time of such verification highly depends on the complexity of the timed-automata model. Therefore, we compare the complexity of our timed-automata with that of the FAn methods and the BF method only on the parameters that we believe exist in any timed-automata model of the system.

We modeled each periodic task with two automata. One automaton decides the release time of each job of the task and the other automaton keeps track of the overall time that a job has been executed. Another automaton decides which task has access to the processor based on priorities, in an SPP policy, and available time slots, in a TDMA policy. We use stopwatch automata [10] which allows us to stop the clock that represents the overall execution of a task in case of a task preemption. Analysis of stopwatch automata is based on an over-approximation on the state space. Comparison between the final results of the BF and FAn methods shows no over-approximation in the number of DHs in our case studies though.

Table 4. CCCA system parameters (time in *ms*)

Task	$\tau_i = (C_i, T_i, D_i, O_i)$	P_i	m	k
Braking control (hard)	$(5, 30, 28, 0)$	4	-	-
Collision avoidance (hard)	$(15, 50, 48, 19)$	3	-	-
Engine control ((m,k)-firm)	$(15, 30, 28, 12)$	2	140	170
Display control ((m,k)-firm)	$(25, 50, 48, 7)$	1	140	170

Table 5. Complexity of firmness analysis methods

Met.	Comp. TDMA	Comp. Asyn. SPP	Comp. Syn. SPP
FAn	$O(w)$	$O(MH_{1^+})$	$O(MT_{hst})$
BF	$O(kwf)$	$O(kH_{1^+}^2 Mf)$	$O(kH^3M^2f)$
TA	$O(k^2 N_{slots}^{c_1})$	$O(k^2 H_{1^+}^3 M^{c_2})$	$O(k^2 H^{c_3} M^{c_4})$
MILP	-	-	$O(k^{c_5} HM^2)$

8.4 Experimental Setup

We took 50 different setups for each scheduling policy. Each setup consists of multiple periodic tasks and a scheduling policy under which the tasks are running. The parameters for our experiments are inspired by the Cruise Control with Collision Avoidance (CCCA) system of [16] [4]. This system consists of two hard real-time tasks, i.e. Braking Control (BC) and Collision Avoidance (CA), and two (m,k)-firm tasks, i.e. Engine Control (EC) and Display Control (DC). Table 4 shows the parameters of these tasks. We took multiple instances of one task when we required to have a combination of more than four tasks. In our experimental setups $k \in \{10, 50, 100, 170\}$ and $M \in \{4, 6, 10, 20, 50\}$. In the cases of TDMA policy, the time wheel size is uniformly chosen in the range $w \in [1, 50]ms$. Moreover, we choose the activation period of tasks in a task set to obtain a wide range of hyperperiod of the tasks, i.e. the hyperperiod of all tasks H and the hyperperiod of the tasks with higher priorities than that of the task under the firmness analysis H_{1^+}. We implemented the BF method and the FAn methods in the C++ programming languages. All the experiments were run on a system with a $2.6GHz$ quad core CPU and 4GB RAM.

8.5 Scalability Analysis and Accuracy

Table 5 compares the complexity of the four methods that are considered in this work obtained from our experiments. In order to obtain the complexity of different methods, we conducted experiments by varying one of the parameters affecting the run-time of the methods while assuming constant value for the rest of the parameters in the range that is mentioned in Section. 8.4.

The run time of the FAn methods mostly depends on the process of obtaining hit zones. This process depends on the period of the hit zone function. The period of the hit zone function equals the length of a time wheel w in a TDMA policy. This justifies that the FAn method for the TDMA policy has complexity $O(w)$. The period of the hit zone function equals the hyperperiod H_{1^+} in the asynchronous SPP policy and the period of the task with the highest priority T_{hst} in the synchronous SPP policy. Moreover, the run time of obtaining the hit zone function in asynchronous and synchronous tasks scales with the number of tasks M as it influences the number of times that Eq. 9 and Eq. 14 are calculated, respectively. Therefore, the FAn methods for asynchronous and synchronous SPP policies have complexity $O(MH_{1^+})$ and $O(MT_{hst})$, respectively.

The process of calculating hit zones exists in the BF method too. Moreover, in the TDMA policy, the run time of the BF method depends on the number of clock cycles in one period of the TDMA time wheel. Therefore, the complexity of the BF method for the TDMA policy is $O(kwf)$ where f is the clock frequency of the underlying operating system. In the asynchronous SPP policy, the number of all possible offsets of a window of k consecutive jobs of τ_1 scales with M and H_{1^+}. Moreover, the run-time of the calculation of the hit zone function depends on H_{1^+} as the number of jobs in one hyperperiod H_{1^+} depends on the length of the hyperperiod. Thus, this approach has complexity $O(kH_{1^+}^2Mf)$. In the synchronous SPP policy, all possible alignments between all tasks have to be considered. Therefore, the hyperperiod of all tasks H and the number of tasks M has bigger impact on the number of possible release times of all tasks than that of an asynchronous SPP policy. The BF method for synchronous SPP policy has complexity $O(kH^3M^2f)$.

The run-time of the timed-automata model depends on the size of the state space built during the reachibility analysis. We believe that in any timed-automata model of the TDMA policy, switching between different time slots and releasing a job of a task are considered as a new state. Our experiments show that the TA method for TDMA has complexity $O(k^2N_{slots}^{c_1})$ where N_{slots} is the number of time slots in one TDMA time wheel and $c_1 \geq 3$. In a timed-automata model of asynchronous and synchronous SPP policies, the size of the state space depends on the number of task switches in hyperperiod H_{1^+} and H, respectively. Our experiments show that the complexity of the TA method for synchronous and asynchronous SPP policies has complexity $O(k^2H_{1^+}^3M^{c_2})$ and $O(k^2H^{c_3}M^{c_4})$, respectively, with $c_2 \geq 2$, $c_3 \geq 3$ and $c_4 \geq 2$.

In the range of parameters that we considered in our experiments, the run-time of the MILP algorithm has complexity $O(k^{c_5}HM^2)$ where $c_5 \geq 3$. Our experiments confirm the fact that the number k is a dominant factor in the complexity of the MILP-based method.

Table 6 shows the results of the TA, FAn and BF methods for three of our experiments out of 50 experiments running under TDMA and asynchronous SPP policies. In all the experiments in Table 6, τ_1 is under firmness analysis. We show case studies with different numbers of tasks M and various hyperperiod H_{1^+} to illustrate their effect on the run-time of the three methods above. Π_1 and Π_2 have the same number of tasks $M = 4$ and different hyperperiod of the tasks with higher priorities than τ_1, i.e. H_{1^+}. Π_1 and Π_3 have hyperperiod H_{1^+} in the same range whilst different number of tasks, i.e. $M = 4$ for Π_1 and $M = 6$ for Π_3. Note that, comparing the run-time of Π_2 and Π_3 is not useful for any method since more than one parameter influencing the scalability of the methods, i.e. M and H_{1^+}, differ between these two task sets. In the case of a SPP policy, the priority of the tasks in each task set is in the same order as the index of the tasks names, e.g. τ_4 and τ_1 have the highest and the lowest priority in Π_1, respectively. In the case of a TDMA policy, one time slot is allocated to each task. Therefore, the number of time slots of the time wheel is equal to the number of the tasks in a task set. The table confirms that the run-time of the TA method is more affected by change in the number of tasks than that of the FAn and the BF method. Moreover, the run time of the BF and the TA method is more affected by H_{1^+} than that of the FAn method (see Table. 5). Table 7 shows the results of firmness analysis of the same task sets as in Table 6 using TA, FAn, BF and MILP-based methods running under a synchronous SPP policy. Using the MILP-based method, none of the three experiments terminated within an hour limit. We later show that the MILP-based method does not scale for the cases with high value of k, e.g. $k = 170$. While the run time of the BF method in some setups is not problematic for design time analysis, the FAn methods may still be preferred when the design of system parameters, e.g. allocated slots in a TDMA policy and priorities in SPP policies, is done in several iterations. In such a process, each design iteration requires a firmness analysis. We define the error of our results (for synchronous SPP policy) as the absolute difference between the minimum DHs obtained by the FAn method and the minimum DHs

Table 6. Three examples of 50 experimental setups considered in this work running under TDMA and asynchronous SPP policies

Task set (τ_1 with (150, 170)-firm condition is under firmness analysis)	H_{1^+}	TDMA				Asyn. SPP				
		DHs	run time			O_{max}	DHs	run time		
			FAn	TA	BF			FAn	TA	BF
$\Pi_1 = \{\tau_4 = (5, 50, 48, 0), \tau_3 = (7, 50, 47, 12), \tau_2 = (12, 30, 30, 19), \tau_1 = (17, 57, 55, O_1)\}$	150	153	< 0.1s	410s	40s	11.88	164	< 0.1s	50s	30s
$\Pi_2 = \{\tau_4 = (5, 50, 48, 0), \tau_3 = (7, 47, 46, 12), \tau_2 = (11, 29, 28, 19), \tau_1 = (16, 57, 55, O_1)\}$	68150	168	11s	16m	22m	41530	161	6s	>1h	40m
$\Pi_3 = \{\tau_6 = (5, 50, 58, 0), \tau_5 = (5, 50, 56, 12), \tau_4 = (7, 30, 28, 19), \tau_3 = (7, 50, 50, 37), \tau_2 = (5, 20, 17, 21), \tau_1 = (13, 57, 55, O_1)\}$	300	138	<1s	5h	50m	231.08	157	5s	>1h	>1h

Table 7. Three examples of 50 experimental setups considered in this work running under synchronous SPP policy

Task set (same as Table 6)	H_{1^+}	DHs	Syn. SPP run time			
			FAn (accuracy)	TA	BF	MILP
Π_1	150	149	< 0.1s (92%)	40m	610s	>1h
Π_2	68150	162	21s (85%)	55m	>1h	>1h
Π_3	300	138	<1s (86%)	>1h	>1h	>1h

obtained by the BF method divided by the minimum DHs obtained by the BF method. Therefore, we obtain the accuracy of the FAn method as follows

$$Acc_{FAn} = 1 - \frac{min_{BF}(DHs) - min_{FAn}(DHs)}{min_{BF}(DHs)} \tag{18}$$

where $min_{BF}(DHs)$ and $min_{FAn}(DHs)$ are the minimum DHs obtained by the BF and the FAn methods, respectively. The average accuracy of the FAn method for all 50 synchronous task sets is 87%. The accuracy of the MILP-based method can be calculated using Eq. 18 by substituting $min_{MILP}(DHs)$, i.e. the minimum number of DHs obtained by the MILP-based method, with $min_{FAn}(DHs)$. The MILP-based method is 97% accurate on an average in those of our experiments which are terminated within the time limit. The accuracy of FAn and MILP methods is influenced by the nature of the task set. For example, the source of inaccuracy (or over-approximation) in the FAn method is approximation performed in Eq. 17. For both methods, the runtime and accuracy are not directly related.

Fig. 9 compares the run-time of the four firmness analysis methods on 50 synchronous task sets with $M = 5$ and three values of $k \in \{10, 50, 100\}$ running under the SPP policy. The FAn method terminates within 35s for all the experiments. Moreover, Fig. 10 depicts the run-time of

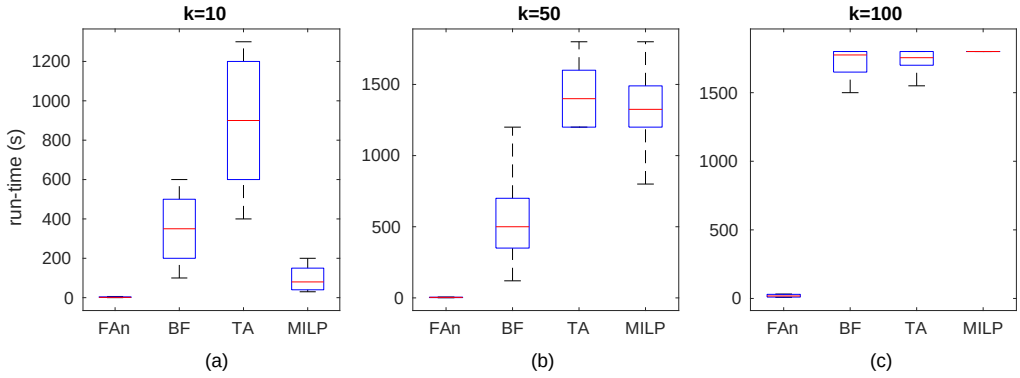

Fig. 9. Run-time comparision among different firmness analysis methods with $M = 5$ and (a) $k = 10$, (b) $k = 50$, (c) $k = 100$.

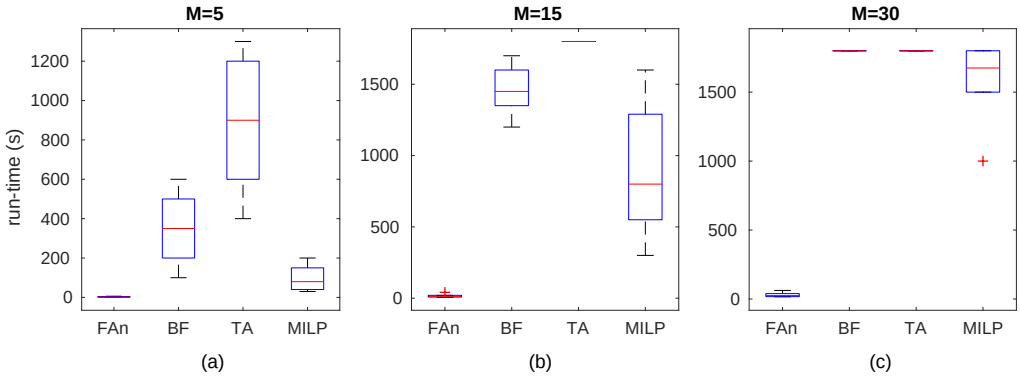

Fig. 10. Run-time comparision among different firmness analysis methods with $k = 10$ and (a) $M = 5$, (b) $M = 15$, (c) $M = 30$.

these methods for 50 other combinations of tasks with $k = 10$ and $M \in \{5, 15, 30\}$. Fig. 9 and Fig. 10 show that the application of the FAn method becomes more crucial in the cases with a high number of tasks M and a large value of k. The MILP-based method may be applied for the cases with a lower k and not very high number of tasks, i.e. $M \leq 15$ in our experiments. The BF method might be used in the cases where the period of the hit zone function is short. The number of tasks M is the dominant factor with respect to the complexity of the TA method. Although the majority of our experiments with $M < 6$ terminated using Uppaal verification, none of the experiments with $M > 8$ terminated within one hour. However, Uppaal verification is not as much affected by the value of k as it is by M.

9 CONCLUSIONS

The proposed generalization of the FAn method provides a framework, based on the Balloon and Rake problem, that allows to verify the satisfaction of firmness conditions under various scheduling policies. Using this framework, we proposed a novel firmness analysis method for the synchronous SPP policy. The proposed method is a foundation for scheduling with deadline misses within

firmness conditions. This allows us to consider better-than-the-worst-case design for real-time tasks as opposed to traditional design with hard deadlines (which is a special case of firmness analysis). One possible future work is to analyse the impact of designing with deadline misses on resource dimensioning.

ACKNOWLEDGMENT

This research was supported by the ARTEMIS joint undertaking under the ALMARVI project (621439).

REFERENCES

[1] Benny Akesson, Anna Minaeva, Premysl Sucha, Andrew Nelson, and Zdenek Hanzalek. 2015. An efficient configuration methodology for time-division multiplexed single resources. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), Proceedings of*. IEEE.

[2] R. Alur and D.L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235.

[3] A. R. B. Behrouzian, D. Goswami, and T. Basten. 2018. Robust co-synthesis of embedded control systems with occasional deadline misses. In *International Symposium on On-Line Testing and Robust System Design (IOLTS)*.

[4] A. R. B. Behrouzian, D Goswami, T Basten, M Geilen, H Alizadeh Ara, and M Hendriks. 2018. Firmness analysis of real-time applications under static-priority preemptive scheduling. In *Real-Time and Embedded Technology and Applcations Symposium (RTAS), Proceedings of*.

[5] A. R. B. Behrouzian, D Goswami, M Geilen, M Hendriks, H Alizadeh Ara, EP van Horssen, WPMH Heemels, and T Basten. 2016. Sample-drop firmness analysis of TDMA-scheduled control applications. In *Industrial Embedded Systems (SIES), International Symposium on*.

[6] G. Bernat. 1998. *Specification and analysis of weakly hard real-time systems*. Universitat de les Illes Balears, Spain.

[7] G. Bernat, A. Burns, and A. Llamosi. 2001. Weakly Hard Real-Time Systems. *Computers, IEEE Transactions on* 50, 4 (2001), 308–321.

[8] T. Bund and F. Slomka. 2015. Worst-case performance validation of safety-critical control systems with dropped samples. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*.

[9] G. Buttazzo. 2011. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media.

[10] F. Cassez and K. Larsen. 2000 United Kingdom. The impressive power of stopwatches. In *Concurrency Theory, International Conference on*. Springer.

[11] M. Coutinho, J. Rufino, and C. Almeida. 2008. Response time analysis of asynchronous periodic and sporadic tasks sheduled by priority preemptive algorithm. In *Real-Time Systems (ECRTS), Euromicro Conference on*.

[12] M. E. M. Ben GaÃŕd, D. Simon, and O. Sename. 2008. Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses. In *IFAC Proceedings*.

[13] D. Goswami, R. Schneider, and S. Chakraborty. 2014. Relaxing Signal Delay Constraints in Distributed Embedded Controllers. *IEEE Transaction on Control Systems Technology* 22, 6 (2014), 2337–2345.

[14] M. Hamdaoui and P. Ramanathan. 1995. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *Computers, IEEE Transactions on* 44, 12 (1995), 1443–1451.

[15] Z. A. H. Hammadeh, S. Quinton, and R. Ernst. 2014. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *Proceedings of International Conference on Embedded Software (EMSOFT)*.

[16] H. Ismail, D. Jawawi, and M. Isa. 2015. A weakly hard real-time tasks on global scheduling of multiprocessor systems. In *Software Engineering Conference (MySEC), Proceedings of*.

[17] Y. Kong and H. Cho. 2011. Guaranteed scheduling for (m, k)-firm deadline-constrained real-time tasks on multi-processors. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), International Conference on*.

[18] J.Y.T. Leung and J. Whitehead. 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation* 2, 4 (1982), 237âĂŞ250.

[19] C. L. Liu and J. W. Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)* 20, 1 (1973), 46âĂŞ61.

[20] J. Ning, Y. Song, and L. Rui-Zhong. 2005. Analysis of networked control system with packet drops governed by (m, k)-firm constraint. In *Fieldbus Systems and Their Applications, International Conference on*.

[21] P. Pazzaglia, L. Pannocchi, A. Biondi, and M. Di Natale. 2018. Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses. In *Euromicro Conference on Real-Time Systems (ECRTS)*.

[22] S. Quinton, T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst. 2014. Typical worst case response-time analysis and its use in automotive network design. In *Design Automation, Proceedings of the 51st Annual Conference on*. ACM.

[23] Y. Sun and M. Natale. 2017. Weakly Hard Schedulability Analysis for Fixed Priority Scheduling of Periodic Real-Time Tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 171.

[24] S. Tobuschat, R. Ernst, A. Hamann, and D. Ziegenbein. 2016. System-level timing feasibility test for cyber-physical automotive systems. In *Industrial Embedded Systems (SIES), International symposium on*.

[25] E. van Horssen, A. Behrouzian, D. Goswami, D. Antunes, T. Basten, and M. Heemels. 2016. Performance analysis and controller improvement for linear systems with (m, k)-firm data losses. In *European Control Conference (ECC)*.

[26] W. Xu, Z. Hammadeh, A. Kröller, R. Ernst, and S. Quinton. 2015. Improved deadline miss models for real-time systems using typical worst-case analysis. In *Real-Time Systems (ECRTS), 27th Euromicro Conference on*.

[27] Q. Zhou, G. Li, and J. Li. 2017. Improved Carry-in Workload Estimation for Global Multiprocessor Scheduling. *IEEE Trans. Parallel Distributed Systems* 28, 9 (2017), 2527–2538.