Hybrid Self-Suspension Models in Real-Time Embedded Systems

Georg von der Brüggen, Wen-Hung Huang, and Jian-Jia Chen
Department of Informatics, TU Dortmund University, Germany

***Abstract*—[1] To tackle the unavoidable self-suspension behavior due to I/O-intensive interactions, multi-core processors, computation offloading systems with coprocessors, etc., the dynamic and the segmented self-suspension sporadic task models have been widely used in the literature. We propose new self-suspension models that are hybrids of the dynamic and the segmented models. Those hybrid models are capable of exploiting knowledge about execution paths, potentially reducing modelling pessimism. In addition, we provide the corresponding schedulability analysis under fixed-relative-deadline (FRD) scheduling and explain how the state-of-the-art FRD scheduling strategy can be applied. Empirically, these hybrid approaches are shown to be effective with regards to the number of schedulable task sets.**

I. Introduction

To ensure safe operations of hard real-time embedded systems, worst-case timeliness needs to be verified, i.e., task instances always have to finish before a certain deadline. Since most real-time embedded systems require *periodic* or *regular* sampling and control of the physical plant, the periodic task model and the sporadic task model are widely used in the literature as the basic system models. In such models, a task is an infinite sequence of task instances, referred to as *jobs*, where two consecutive jobs of a task should arrive exactly periodically in the periodic task model or with a minimum inter-arrival time separation in the sporadic task model. Extensive research results have been obtained for both task models and a verity of different settings has been considered. One assumption of the majority of analyses and scheduling algorithms is that a job does not voluntarily suspend its execution. However, self-suspension behavior is widely spread in many real-time applications due to unavoidable 1) interactions with external devices and accelerators [14], 2) resource competitions in multicore systems with shared resources [13], 3) suspension-aware multiprocessor synchronization protocols [4], etc. Although the impact of self-suspension behavior in real-time systems has been investigated since 1990, the literature of this research topic has been seriously flawed as reported in [8].

Two concrete models have been studied in the literature: the *dynamic* and the *segmented* self-suspension (sporadic) task model.[2] In the dynamic self-suspension model a task τ_i is specified like an ordinary sporadic task that has its worst-case self-suspension time S_i as an additional parameter. A job of task τ_i can suspend itself at any moment and several times if needed before it finishes as long as the total self-suspension time of the job is not more than S_i. By contrast, the *segmented* self-suspension model defines an interleaved execution and self-suspension pattern for any job of a task τ_i that is composed of $m_i + 1$ computation segments separated by m_i suspension intervals, where $C_{i,j}$ is the worst-case execution time of a computation segment, and $S_{i,j}$ is the worst-case length of a self-suspension interval, i.e., the execution/ suspension pattern is $(C_{i,1}, S_{i,1}, C_{i,2}, S_{i,2}, ..., S_{i,m_i}, C_{i,m_i+1})$. These two models are applicable in different scenarios with a high tradeoff between flexibility and accuracy:

- The dynamic self-suspension model can be used when only limited information about the suspension behavior is known. It has higher flexibility at the sacrifice of pessimistic analyses and designs of scheduling policies if the suspending pattern can be defined precisely. Specifically, this model has been studied in [5], [7], [11], [15].
- The segmented self-suspension model has lower flexibility, but the self-suspending structure can be exploited by the scheduling algorithms to make better decisions. However, such a concrete segmented pattern is only achievable if the structure of the program is well designed and the execution pattern is determinable. Specifically, this model has been studied in [5], [6], [12], [17]–[21].

A well-known approach to schedule a set of segmented self-suspension tasks is the fixed-relative deadline (FRD) scheduling policy, originally proposed by Chen and Liu [6]. An FRD approach assigns an individual relative deadline to each subtask, i.e., to each computation segment, and all subtasks are scheduled according to Earliest Deadline First (EDF). When each task only has one self-suspension interval, the state-of-the-art FRD deadline assignment policy is the Shortest Execution Interval First Deadline Assignment (SEIFDA), proposed by von der Brüggen et al. [21]. For a detailed review of self-suspension please refer to [8].

As mentioned before, the majority of the literature assumes either dynamic or segmented self-suspension patterns. On the one hand, the dynamic self-suspension model is very flexible but inaccurate. On the other hand, the segmented self-suspension model is very restrictive but very accurate. There is a big gap between these two widely-adopted self-suspension task models, which can be potentially filled by the *hybrid self-suspension task models* proposed in this paper.

Contributions: In this paper, we propose several *hybrid self-suspension task models* that are more flexible than the segmented self-suspension task model and less pessimistic than the dynamic self-suspension task model, therefore achieving different levels of tradeoff between flexibility and precision.

- The proposed hybrid self-suspension task models are specified in Section II. Compared to the spo-

[1]This paper is supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/).

[2]One exception is Bletsas' dissertation [3], where a *directed acyclic graph* (DAG) was used to represent the task control flow and reduced to the segmented or dynamic self-suspension model. However, the DAG self-suspension model by Bletsas [3] has never been directly used in the literature to make better scheduling decisions or to improve scheduling analyses.

radic task model, the hybrid models have an additional parameter m_i predefining the number of self-suspension intervals. However, instead of using $(C_{i,1}, S_{i,1}, C_{i,2}, S_{i,2}, ..., S_{i,m_i}, C_{i,m_i+1})$ as the concrete execution/suspension pattern, the hybrid self-suspension task models provide several options to model the tasks, depending on whether the execution/suspension pattern of a job can be known when it arrives to the system or not:
    - *Pattern-oblivious Models*: The execution pattern of a job is unknown. This model only assumes that the number of self-suspension intervals of a job of task τ_i is at most m_i. However, all possible execution paths may be known offline. We specifically explore two cases:
        - *Individual Upper Bounds:* Bounded execution time of the *j-th* computation segment.
        - *Multiple Paths:* Each task τ_i is described by a set of p patterns of worst-case execution times and maximum suspension times where each pattern describes a possible execution path.
    - *Pattern-clairvoyant Model*: The individual execution/suspension pattern of each job is of τ_i is known the moment the job arrives.
- We show how these models can be applied by carefully examining the special case that each task has only one self-suspension interval, i.e., $m_i = 1$. The applicability of FRD and extensions of SEIFDA for the different hybrid self-suspension task models are provided in Section IV. For completeness, the FRD scheduling policy, related schedulability tests using Demand Bound Functions (DBF), and SEIFDA are explained in Sec. III.
- Empirically, these approaches are shown effective in terms of the number of task sets that are schedulable in Section V. Compared to the dynamic self-suspension task model and the segmented self-suspension task model (that enforces the execution upper bounds on the computation segments), the hybrid self-suspension task models can achieve different degrees of improvement, depending on the properties of the execution/suspension patterns.

Although we focus on one self-suspension interval in our analyses, Huang and Chen [12] already showed that FRD is also a valid approach for multiple self-suspension intervals. We strongly believe that the results in this paper open a new dimension for suspension-aware real-time embedded systems. For example, in the past, the dynamic self-suspension task model has been widely used for analyzing the multiprocessor synchronization protocols, e.g., [4]. If the number of suspension intervals is small, our conclusion shows that quantifying the execution/suspension patterns can potentially help to improve the schedulability significantly.

II. The Hybrid Self-suspension Task Models

In this section we explain the basic task models and terminology used in this paper and introduce the different *hybrid self-suspension task models* we propose.

We consider n independent sporadic self-suspending real-time tasks $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ in a uniprocessor system. Each task can release an infinite number of jobs (or task instances) under a given minimum inter-arrival time (temporal) constraints T_i, also called the tasks period. The relative deadline of a task τ_i is denoted as D_i and the Worst-Case Execution Time (WCET) is denoted as C_i. This means if a job of task τ_i arrives at time θ_a, 1) the job should finish C_i time units of execution by $\theta_a + D_i$ and 2) the next instance of the task must arrive not earlier than $\theta_a + T_i$. In this work, we restrict our attention to *implicit-deadline* systems, i.e., $D_i = T_i$.

In addition to T_i, D_i, and C_i, in self-suspension task models the maximum suspension time S_i is introduced as an additional parameter. If task τ_i may suspend itself, then S_i is positive; otherwise S_i is 0, i.e., the task is an ordinary sporadic task. In the dynamic self-suspension model, a job of task τ_i can suspend its execution at any moment as long as the total self-suspension time of the job is not more than S_i, i.e., the concrete execution/suspension pattern is unknown. In the *segmented* self-suspension model a task τ_i is defined by a concrete execution/suspension pattern $(C_{i,1}, S_{i,1}, C_{i,2}, S_{i,2}, ..., S_{i,m_i}, C_{i,m_i+1})$, i.e., any job of τ_i is composed of $m_i + 1$ computation segments that are separated by m_i suspension intervals. $S_{i,j}$ is the worst-case length of the j-th self-suspension interval while $C_{i,k}$ is the WCET of the k-th computation segment for a segmented self-suspension task τ_i. The total WCET $C_i = \sum_{j=1}^{m_i+1} C_{i,j}$ and the total self-suspension time $S_i = \sum_{j=1}^{m_i} S_{i,j}$ in the segmented model.

In the *hybrid self-suspension task models*, we assume that in addition to S_i the number of self-suspension intervals m_i is known as well for each task. This means that the execution of each job of τ_i is composed of at most $m_i + 1$ *computation* segments separated by m_i *suspension* intervals, similar to the segmented self-suspension model. The sum of the execution times of the computation segments of a job of task τ_i is at most its worst-case execution time C_i, while the sum of the lengths of the self-suspension intervals of a job of task τ_i is at most its worst-case suspension time S_i. All these values are positive for self-suspending tasks. For a non-suspending task S_i and m_i are both 0. These models are

- more precise than the traditional dynamic self-suspension task model, where m_i is not considered, and
- more flexible and less precise than the traditional segmented self-suspension task model, where the worst-case execution time of each of the $m_i + 1$ computation segments and the worst case suspension time for each of the m_i suspension intervals is fixed and specified.

For the rest of this paper, we implicitly call such tasks as self-suspending tasks if the context is clear. As we will adopt the Fixed-Relative-Deadline strategy [6], each subtask will be assigned to its own relative deadline. All subtasks are scheduled according to Earliest Deadline First (EDF). We do not assume that each task in the task set must be a self-suspending task. If a task has no self-suspension behavior, such an ordinary (non-suspending) sporadic task should still be scheduled by using its original deadline. However, for the simplicity of presentation, we do not consider these tasks here.

The hybrid self-suspension task models are applicable under the assumption that each task can be described by a set of p disjunct execution/suspension patterns[3] similar to the patterns used in the segmented self-suspension model. As briefly explained in Section I, the hybrid self-suspension

[3]The pattern-oblivious approach with individual upper bounds can also be applied if only the individual bounds, S_i^{max}, C_i^{max}, and m_i are known but further information about the internal structure of the task is not available.

suspension model	flexibility?	accuracy?
dynamic [7], [11], [15]	very flexible (high)	inaccurate (low), *over flexible*
pattern-oblivious (hybrid, this paper)	less flexible than dynamic (medium to high)	applicable in most cases for known m_i (low to medium)
pattern-clairvoyant (hybrid, this paper)	less flexible than pattern-oblivious (medium to low)	more accurate than patter-oblivious (medium to high)
segmented [6], [12], [17]–[21]	very restrictive (low)	only accurate and applicable for fixed patterns (high), *over restrictive*

TABLE I: High-level comparison of different self-suspension models.

task models provide several options depending on the number of possible execution/suspension patterns, how detailed the informations we can derive for each of these patterns are, and whether the execution pattern of a job can be known when the job arrives to the system or not. Suppose that a job of task τ_i is released at time θ_a. If the (high-level) execution/suspension pattern of the job cannot be identified at time θ_a, such a scenario is called *Pattern-oblivious*. If the pattern can be identified, e.g., by checking (some of) the known input values when a job arrives at time θ_a (potentially with approximations), such a scenario is called *Pattern-clairvoyant*. Clearly, pattern-clairvoyant approaches only work if the overhead for the identification is negligible. We consider the following scenarios:

- *Pattern-oblivious*: Depending on the knowledge of the execution times of the computation segments and the self-suspension time, we consider two cases:
  - *Individual Upper Bounds*: We assume known bounds on the execution time of the *j-th* computation segment, i.e., $C_{i,j}$ is no more than the individually specified $C_{i,j}^{max}$ for each $j = 1, 2, \ldots, m_i + 1$. In addition, the suspension of a job of task τ_i takes place at most m_i times and the j-th suspension is for at most $S_{i,j}^{max}$ amount of time for each $j = 1, 2, \ldots, m_i$. Moreover, we further assume that the worst-case execution time of a job of task τ_i is at most C_i^{max} while the maximum suspension time is at most S_i^{max}. Specifically, by the above definition $\sum_{j=1}^{m_i+1} C_{i,j}^{max} \geq C_i^{max}$ and $\sum_{j=1}^{m_i} S_{i,j}^{max} \geq S_i$. This scenario also covers a special case when $C_{i,j}^{max}$ is set to C_i^{max} for each j, i.e., no specific segmented information is revealed.
  - *Multiple Paths*: A task τ_i is described by p different execution paths with known execution/suspension patterns, in which a job of task τ_i can suspend at most m_i times. For the special case $m_i = 1$ this results in a set of p triples: $\{(C_{i,1}^1, S_i^1, C_{i,2}^1), \ldots, (C_{i,1}^p, S_i^p, C_{i,2}^p)\}$. Note that we know all possible paths, but the system is not able to identify which path will be executed during runtime, i.e., at the moment a job arrives.
- *Pattern-clairvoyant*: Each job of task τ_i has its individual execution/suspension pattern and the pattern that will be executed is known at the beginning of a job. Note that different jobs of a task may have different execution/suspension patterns. As mentioned earlier, we assume such an identification takes negligible time.[4] This is more precise than the two models above. If all the jobs of task τ_i have the same pattern, then the model becomes the segmented self-suspension task model.

Table I provides a summary of the flexibility and the accuracy of different self-suspension task models.

The utilization of task τ_i is defined as $U_i = C_i/T_i$. We implicitly assume that $C_i + S_i \leq T_i$. We use the following definitions of feasibility and schedulability in this paper:

- A schedule is *feasible* if there is no deadline miss and all the timing constraints are respected.
- A task system $\mathbf{T}$ is called *schedulable* if there exists a feasible schedule for the task system for any release patterns under the temporal constraints.
- A task system $\mathbf{T}$ is *schedulable* under a scheduling algorithm if the schedule produced by the algorithm for the task system is always feasible.

III. Introduction of FRD Strategies

When a Fixed-Relative-Deadline (FRD) strategy [6], [12], [19], [21] is adopted, for each $\tau_i \in \mathbf{T}$ an individual relative deadline $D_{i,j}$ is assigned for the j-th computation segment of task τ_i. When a job of task τ_i arrives at time t,

- its first subjob (i.e., its first computation segment) arrives at time t and has its absolute deadline set to $t + D_{i,1}$,
- its first self-suspension has to be finished before $t + D_{i,1} + S_{i,1}$, and
- its j-th subjob, with $j \geq 2$, (i.e., its j-th computation segment) arrives at the time when the $(j-1)$-th self-suspension interval finishes and has its absolute deadline at $t + \sum_{\ell=1}^{j-1}(D_{i,\ell} + S_{i,\ell}) + D_{i,j}$ for $j = 2, 3, \ldots, m_i + 1$.[5]

All subjobs of the tasks are scheduled according to the well-known Earliest-Deadline-First (EDF) scheduling strategy, i.e., the subjob in the ready queue with the earliest absolute deadline has the highest priority. *We will focus on a special case in which each task has at most one self-suspension interval, i.e., $m_i = 1$ for every τ_i.*

The general schedulability tests for FRD strategies under the segmented self-suspension model were presented by von der Brüggen et al. in [21]. As we adopt those scheduling tests here, we will briefly introduce the Demand Bound Functions (DBF) and schedulability tests presented in [21] now. The DBFs are used to calculate the maximum possible interference of a task over a given interval $[t_0,\ t_0 + t]$ where $t_0 = 0$ can be implicitly assumed for the simplicity of presentation.

Since we will use EDF for scheduling the subjobs, we recall the following property derived by Chetto and Chetto [9].

Lemma 1 (Chetto and Chetto [9]). *We are given a set $\mathbf{J}$ of jobs, in which each job J_j has its arrival time a_j, worst-case*

[4]It is also possible to include the time for identifying the execution pattern of the job as part of the first computation segment(s) in all the paths. For such a case, our analysis has to be revised and adjusted to give the identification process the highest priority.

[5]The FRD here is a non-enforced FRD. Note that it has been already explained in [6], [21] that releasing its j-th subjob at the moment when its $j-1$-th self-suspension interval finishes (instead when $t + \sum_{\ell=1}^{j-1}(D_{i,\ell} + S_{i,\ell})$) does not change the schedulability condition, as the subjobs are scheduled using EDF.

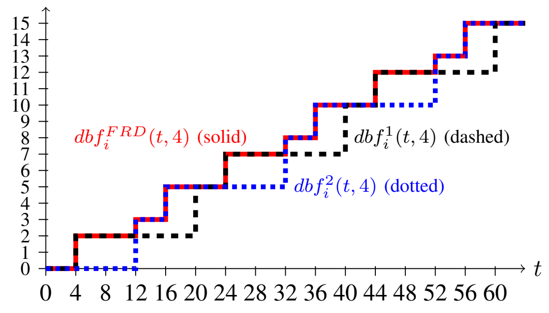

Fig. 1: An example of $dbf_i^{FRD}(t, D_{i,1})$ (red, solid) and the functions $dbf_i^1(t, D_{i,1})$ (gray, dashed) and $dbf_i^2(t, D_{i,1})$ (blue, dotted) it is composed of, for $D_{i,1} = 4$, $C_{i,1} = 2$, $C_{i,2} = 3$, $S_i = 4$, and $T_i = 20$, which leads to $D_{i,2} = 12$.

execution time C_j and, absolute deadline d_j. The set **J** *can meet the deadlines on one processor by using EDF, if and only if the following condition holds:*

$$\forall a_i < d_k, \quad \sum_{\tau_j: a_i \le a_j \text{ and } d_j \le d_k} C_j \le d_k - a_i \tag{1}$$

The concept of the DBF can be explained by using Eq. (1). For a task τ_i in a given interval $[t_0, t_0+t]$, i.e., a_i is t_0 and d_k is t_0+t in Eq. (1), its maximum contribution to the left-hand side of Eq. (1) is due to the jobs arrived at or after t_0 with absolute deadlines at or before t_0+t. Regardless of the considered task models, as long as (working conserving) EDF scheduling is applied, the schedulability analysis is simply to first quantify $dbf_i^{\mathcal{A}}(t)$ as the maximum demand requested by a task τ_i in any given interval $[t_0, t_0+t]$, and then to validate whether $\sum_{\tau_i} dbf_i^{\mathcal{A}}(t) \le t$ for every $t \ge 0$, where $\mathcal{A}$ is a specific algorithm for deciding the relative deadline assignments.

A. Schedulability Tests for FRD Scheduling

One intuitive way to develop the DBFs of a task under an FRD scheduling policy is to represent the task as a generalized multiframe (GMF) task [1]. We explain the case with one self-suspension interval for each task but the approach can be easily extended to cover multiple self-suspension intervals.

In general, a multiframe task is described by a 3-tuple of vectors with identical length $(\overrightarrow{C_i}, \overrightarrow{D_i}, \overrightarrow{T_i})$ representing the WCETs, relative deadlines, and interarrival times of the frames, respectively. For a one-segment self-suspending task, there are two frames in the GMF task model: $\tau_i = \{(C_{i,1}, D_i^1, T_i^1), (C_{i,2}, D_i^2, T_i^2)\}$.[6] The j-th frame of a task τ_i has the WCET, relative deadline, and interarrival time of the $(j \mod 2)$-th frame. As the second computation segment is released after the suspension interval we know that $D_i^1 = D_{i,1}$ and $T_i^1 = D_{i,1} + S_i$. Moreover, $T_i^2 = T_i - T_i^1 = T_i - D_{i,1} - S_i$ and $D_i^2 = D_{i,2} = T_i - D_{i,1} - S_i$. To get the maximum demand over an interval $[t_0, t_0 + t]$ for a GMF task one of the frames must be released at time t_0 and all consecutive frames are released as early as possible.

If $C_{i,1}$ is released at t_0 it has to be finished not later than $t_0 + D_{i,1}$. $C_{i,2}$ is released at $D_{i,1} + S_i$ and has to be finished at $t_0 + D_{i,1} + S_i + D_{i,2} = T_i$. This pattern repeats periodically with period T_i. As shown in [21] this leads to

$$dbf_i^1(t, D_{i,1}) = \left\lfloor \frac{t + (T_i - D_{i,1})}{T_i} \right\rfloor C_{i,1} + \left\lfloor \frac{t}{T_i} \right\rfloor C_{i,2} \tag{2}$$

When $C_{i,2}$ is released at t_0 it has to be finished at $t_0 + D_{i,2}$. $C_{i,1}$ is released at $t_0 + D_{i,2}$ and has to be finished at $t_0 + D_{i,1} + D_{i,2}$. Therefore, the resulting DBF is

$$dbf_i^2(t, D_{i,1}) = \left\lfloor \frac{t + (D_{i,1} + S_i)}{T_i} \right\rfloor C_{i,2} + \left\lfloor \frac{t + S_i}{T_i} \right\rfloor C_{i,1} \tag{3}$$

Lemma 2. *The DBF for τ_i under an FRD assignment is the maximum of the two patterns:*

$$dbf_i^{FRD}(t, D_{i,1}) = \max(dbf_i^1(t, D_{i,1}), dbf_i^2(t, D_{i,1})) \tag{4}$$

Proof: This follows from Eqs. (2) and (3). ∎

An example that shows how $dbf_i^1(t, D_{i,1})$, $dbf_i^2(t, D_{i,1})$ and the resulting $dbf_i^{FRD}(t, D_{i,1})$ are constructed for given values of $C_{i,1}$, $C_{i,2}$, S_i, T_i, and $D_{i,1}$ can be found in Figure 1.

Lemmas 1 and 2 directly lead to the following exact schedulability test for a given $D_{i,1}$ as presented in [21]:

Theorem 1 (Theorem 1 in [21])**.** *An FRD schedule is feasible if and only if*

$$\sum_{\tau_i \in \mathbf{T}} dbf_i^{FRD}(t, D_{i,1}) \le t, \qquad \forall t \ge 0. \tag{5}$$

We note that in [21] only the segmented self-suspension model is considered. For the hybrid models the DBFs have to be extended to cover all possible cases.

B. Deadline Assignments

The key question for FRD strategies is the assignment of the relative deadlines of the subjobs, i.e., how to distribute the execution interval $T_i - S_i$ among the subjobs. The following approaches have been used in the literature:

- Proportional (Proportional relative deadline assignment): $D_{i,1} = \frac{C_{i,1}}{C_{i,1}+C_{i,2}} \cdot (T_i - S_i)$; $D_{i,2} = \frac{C_{i,2}}{C_{i,1}+C_{i,2}} \cdot (T_i - S_i)$, introduced by Liu et al. [16] in 2014.
- EDA (Equal relative Deadline Assignment): $D_{i,1} = D_{i,2} = (T_i - S_i)/2$, by Chen and Liu [6] in 2014.
- SEIFDA (Shortest Execution Interval First Deadline Assignment): task deadlines are assigned in increasing order of the tasks execution interval $T_i - S_i$, introduced by von der Brüggen et al. [21] in 2016.

While Proportional looks reasonable and EDA looks pessimistic it was shown in [6] that Proportional can perform poorly in the worst case while EDA has a speedup factor of 3 with respect to an optimal algorithm and a speedup factor for 2 compared to an optimal FRD algorithm. Note that EDA can be adopted with no information about the WCET of the subtasks while Proportional can only (directly) be adopted if the WCETs of both subjobs are known. Both Proportional and EDA have in common that the deadline assignment of a task τ_i is independent from the other tasks in the set.

SEIFDA [21] is a greedy approach that assigns the relative deadlines of tasks in increasing order of the tasks execution

[6] Superscripts are used for terms when referring to the GMF task.

interval $T_i - S_i$ with respect to the previously assigned deadlines of tasks with smaller execution interval. The workload provided by these tasks is considered by adding up the related demand bound functions. As SEIFDA assigns the deadlines of the tasks individually, suppose that $T_i - S_i \leq T_k - S_k$ and that $D_{i,1}$ and $D_{i,2}$ with $D_{i,1} + D_{i,2} = T_i - S_i$ are already assigned for $i = 1, 2, \ldots, k-1$. The interesting question is how to assign the deadlines for $D_{k,1}$ and $D_{k,2}$. The general approach is that SEIFDA assigns the deadline $D_{k,1}$ for the shorter computation segment $C_{k,1}$ and $D_{k,2}$ is set accordingly. Note that the assumption that $C_{k,1} \leq C_{k,2}$ is just for the simplicity of presentation. If $C_{k,1} > C_{k,2}$ the two computation segments can be swapped before the assignment and then swapped back with the related deadlines afterwards as shown in [21]. When SEIFDA assigns $D_{k,1}$, first the possible values for $D_{k,1}$ are determined. If x is a possible value for $D_{k,1}$ then

$$dbf_k^{FRD}(t, x) + \sum_{i=1}^{k-1} dbf_i^{FRD}(t, D_{i,1}) \leq t, \;\; \forall t \geq 0 \tag{6}$$

holds. Let $\mathbf{X} = \{x \in [C_{k,1}, (T_k - S_k)/2] \mid \text{x is a valid } D_{k,1}\}$ denote the set of all possible values for the deadline assignment of $D_{k,1}$. The upper bound of $(T_k - S_k)/2$ for the possible deadline values is due to the assumption that $C_{k,1} \leq C_{k,2}$ and the observation that it does not make sense to assign the smaller deadline to the longer execution segment. If $\mathbf{X}$ is $\emptyset$ for a task τ_k the task set is not schedulable by SEIFDA. Otherwise, the assignment of $D_{k,1}$ depends on the chosen deadline assignment strategy:

- minD: $D_{k,1}$ is the minimum x in $\mathbf{X}$.
- maxD: $D_{k,1}$ is the maximum x in $\mathbf{X}$.
- Proportionally-Bounded-Min (denoted by PBminD): $D_{k,1}$ is the minimum x such that x is in $\mathbf{X}$ and $x \geq \frac{C_{k,1}}{C_{k,1}+C_{k,2}} (T_k - S_k)$ holds.

The task set is feasible using FRD under EDF if we can find a feasible deadline assignment for each task in $\tau_i, \ldots, \tau_n$ using the given strategy and the task set is schedulable under the given assignment, i.e., Eq. 5 and therefore Theorem 1 holds.

It was shown in [21] that SEIFDA-maxD strictly dominates EDA and SEIFDA-PBminD strictly dominates the Proportional relative deadline assignment.[7] Therefore, in this paper, we will adopt only SEIFDA in the experiments. Note that while EDA can directly be applied without any information about $C_{k,1}$ and $C_{k,2}$, and only knowing S_k, while SEIFDA needs those information. However, if C_k^{max} is known SEIFDA can be applied using $C_{k,1} = C_{k,2} = C_k^{max}$ if S_k is know.

IV. FRD for Hybrid Self-Suspension

In this section, we present how FRD strategies can be applied for hybrid self-suspension task models when each task has at most one suspension interval. Existing FRD strategies assume the segmented self-suspension model, i.e., $C_{i,j}$ is the WCET of the j-th computation segment of each job of τ_i. This is different from the hybrid self-suspension task models that assume a set of given patterns and not only a single pattern.

Depending on the priorly known or determinable knowledge about $C_{k,1}$, $C_{k,2}$, and S_k, different FRD strategies for the pattern-oblivious scenarios are presented in Sec. IV-A and Sec. IV-B while pattern-clairvoyant scenarios are discussed in Sec. IV-C. To perform the schedulability tests efficiently, approximated DBFs are explained in Sec. IV-D.

As a running example we use a task τ_i with three execution patterns (paths) provided in Table II. The execution patterns are denoted τ_i^1, τ_i^2, and τ_i^3. While the period $T_i = 30$ is identical for all execution patterns, $C_{i,1}$, $C_{i,2}$, $C_i = C_{i,1} + C_{i,2}$, and S_i depend on the execution pattern. We assume that when a job of τ_i arrives one of these execution patterns is executed.

A. Pattern-Oblivious: Individual Upper Bounds

We assume to know individual upper bounds (IUB) of the execution time of each computation segment, i.e., $C_{i,j}^p \leq C_{i,j}^{max}$ for each execution pattern p with $j \in \{1, 2\}$, and the maximum suspension time[8] $S_i^{max} = \max\{S_i^p\}$. Let the maximum total execution time among all patterns be $C_i^{max} = \max\{C_{i,1}^p + C_{i,2}^p \mid \text{p is a possible execution pattern}\}$. Note that to apply this model no explicit knowledge about the individual execution/suspension patterns is needed as long as C_i^{max}, $C_{i,1}^{max}$, $C_{i,2}^{max}$, and S_i^{max} can be determined.

We construct the two resulting DBFs for the case where $C_{i,1}$ is released at t_0 and for the case where $C_{i,2}$ is released at t_0 in Eq. (8) and Eq. (9), respectively. If $C_{i,1}$ be released at t_0 the DBF is periodic with period T_i. C_i^{max} is the workload considered for every period but the last one. Note, that it is possible that there are 0 full periods before the time t that is analyzed. To take care of the workload in a period that has started before the analyzed time t but did not finish at time t, i.e., the last period, we define a function G_i^I that helps us to sum up the workload inside one period for notational brevity:

$$G_i^I(t, D_{i,1}) = \begin{cases} 0 & \text{if } 0 \leq t < D_{i,1} \\ C_{i,1}^{max} & \text{if } D_{i,1} \leq t < T_i \end{cases} \tag{7}$$

Therefore, the corresponding demand bound function is

$$dbf_i^{I,1}(t, D_{i,1}) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i^{max} + G_i^I\left(t - \left\lfloor \frac{t}{T_i} \right\rfloor T_i, D_{i,1}\right) \tag{8}$$

if $C_{i,1}$ is released at t_0. In Eq. (8) the first part determines the maximum execution time of the released jobs for completed periods, i.e., both computation segments are finished, while the second part adds a computation segment $C_{i,1}$ if needed. If the first computation segment of task τ_i released after or at t_0 is from $C_{i,2}$, then we have to consider one $C_{i,2}^{max}$ at $D_{i,2}$ and the first release of $C_{i,1}$ happens at $D_{i,2}$. Therefore, the corresponding demand bound function is

$$dbf_i^{I,2}(t, D_{i,1}) = \begin{cases} 0 & \text{if } 0 \leq t < D_{i,2} \\ C_{i,2}^{max} + dbf_i^{I,1}(t - D_{i,2}, D_{i,1}) & \text{if } t \leq D_{i,2} \end{cases} \tag{9}$$

where $D_{i,2}$ is $T_i - S_i - D_{i,1}$. From the discussion for deriving Eqs. (8) and (9) the DBF directly follows:

Lemma 3. *The DBF of τ_i for the pattern-obvious model with individual upper bounds under an FRD assignment is:*

$$dbf_i^I(t, D_{i,1}) = \max(dbf_i^{I,1}(t, D_{i,1}), dbf_i^{I,2}(t, D_{i,1})) \tag{10}$$

Considering Eq. (10) it is not difficult to show that $D_{i,1}$ should be no more than $(T_i - S_i)/2$ if $C_{i,1}^{max} \leq C_{i,2}^{max}$ and

[7]The dominance of SEIFDA-PBminD over Proportional was not stated in [21]. However, the argumentation is identical to the case that SEIFDA-maxD dominates EDA.

[8]If $m_i > 1$, one would consider $S_{i,j}^{max}$ individually for each suspension interval and S_i^{max} would be defined similar to C_i^{max}.

Given Task Parameters					Hybrid Self-Suspension Model								
					IUB		MP		SSSD		PDAB, Bias 2		
$T_i = 30$	$C_{i,1}$	$C_{i,2}$	C_i	S_i	$D_{i,1}$	$D_{i,2}$	$D_{i,1}$	$D_{i,2}$	$D_{i,1}$	$D_{i,2}$	Ratio	$D_{i,1}$	$D_{i,2}$
τ_i^1	2	3	5	5	8	14	8	17	8	17	10/15	12	13
τ_i^2	4	3	7	8				14	14	8	12.6/9.4	11	11
τ_i^3	2	7	9	7				15	8	15	5.1/17.9	7.1	15.9
max	4	7	9	8									

TABLE II: Example: τ_i with 3 versions and deadline assignment for the strategies presented from Sec. IV-A to Sec. IV-C.

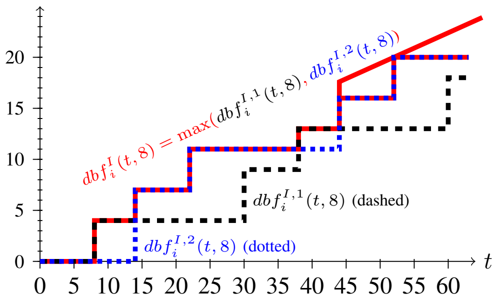

Fig. 2: IUB: dbf_i^I for τ_i in Table II, linear approximation (solid line) for $g = 1$, i.e., after $t = g \cdot T_i + D_{i,2} = 44$.

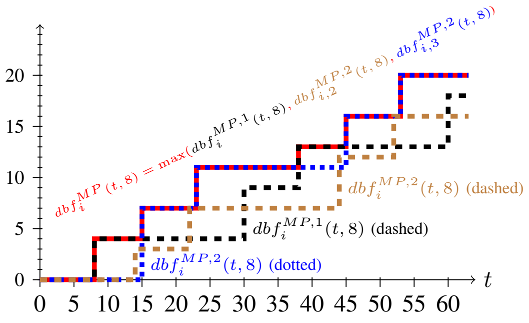

Fig. 3: MP: dbf_i^{MP} for τ_i in Table II. For each path i the DBF when $C_{i,2}$ is released at 0 is considered individually. However, for clarity, $dbf_{i,1}^{MP,2}$ is omitted as it is strictly smaller than $dbf_{i,2}^{MP,2}$.

vice versa. Therefore, we can apply SEIFDA. However, we should note that the assignment strategies in SEIFDA focuses on $D_{i,1}$ when $C_{i,1}^{max} \leq C_{i,2}^{max}$ and $D_{i,2}$ if $C_{i,1}^{max} > C_{i,2}^{max}$.

Figure 2 shows the above functions for task τ_i as listed in Table II. $C_{i,1}^{max}$, $C_{i,2}^{max}$, C_i^{max} and S_i^{max} are calculated as the maximum of the 3 execution/suspension patterns. As these values are independent from the deadline assignment, they are listed as *Given Task Parameters* in Table II. For IUB the value of $D_{i,1}$ is chosen using SEIFDA under the given strategy and $D_{i,2}$ is set accordingly using to $D_{i,2} = T_i - S_i^{max} - D_{i,1}$. For the example in Table II using IUB we assume the deadline assignment strategy sets $D_{i,1} = 8$, and, therefore, with $S_i^{max} = 8$ we get $D_{i,2} = 30 - 8 - 8 = 14$. The resulting $dbf_i^{I,1}(t, D_{i,1})$, $dbf_i^{I,2}(t, D_{i,1})$, and $dbf_i^I(t, D_{i,1})$ are shown in Figure 2.

B. Pattern-Oblivious: Multiple Paths

We assume that a task τ_i is described by a set of p triples of worst-case execution times and maximum suspension times $\{(C_{i,1}^1, S_i^1, C_{i,2}^1), \ldots, (C_{i,1}^p, S_i^p, C_{i,2}^p)\}$, each describing a possible execution/suspension pattern. However, when a job arrives in the system at time t, we do not know which path will be executed. When adopting FRD for such a scenario, we will use a fixed $D_{i,1}$ across all the execution paths. Note that the second computation segment of the job always has an absolute deadline of $t + T_i$. If the first computation segment can meet its deadline, then the second computation segment is released at time $t + D_{i,1} + S_i^j$ for the j-th execution path.

For the deadline assignment of τ_i, first $C_{i,1}^{max}$, $C_{i,2}^{max}$, and S_i^{max} are calculated. The actual deadline assignment is based on these values. Especially, they are used to calculate the minimum value for $D_{i,1}$ if PBminD is used. $G_i^{MP}(t, D_{i,1})$ is defined identically to the related function in Eq. (7) for the case when individual upper bounds are used, as $D_{i,1}$ is identical over all execution patterns.

Similar to the analysis in Sec. IV-A, we consider two general cases. If the first computation segment of task τ_i released after or at t_0 is from $C_{i,1}$, then the corresponding demand bound function $dbf_i^{MP,1}(t, D_{i,1})$ is the same as $dbf_i^{I,1}(t, D_{i,1})$ in Eq. (8) as $D_{i,1}$ is identical for all patterns:

$$dbf_i^{MP,1}(t, D_{i,1}) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i^{max} + G_i^{MP}\left(t - \left\lfloor \frac{t}{T_i} \right\rfloor T_i, D_{i,1}\right) \tag{11}$$

While $dbf_i^{MP,1}(t, D_{i,1})$ is independent from the version that is executed as $D_{i,1}$ is the same for each version, the DBF for the case where the second computation segment is released at t_0 depends on the version of the task. If the first computation segment of task τ_i released after or at t_0 is from $C_{i,2}^j$, i.e., from version j, then the corresponding demand bound function is

$$dbf_{i,j}^{MP,2}(t, D_{i,1}) = \begin{cases} 0 & \text{if } 0 \leq t < D_{i,2}^j \\ C_{i,2}^j + dbf_i^{MP,1}(t - D_{i,2}^j, D_{i,1}) & \text{if } t \leq D_{i,2}^j \end{cases} \tag{12}$$

where $D_{i,2}^j$ is $T_i - S_i^j - D_{i,1}$.

Lemma 4. *The DBF of τ_i for the pattern-obvious model with multiple paths under an FRD assignment is as follows:*

$$dbf_i^{MP}(t, D_{i,1}) = \max\left(dbf_i^{MP,1}(t, D_{i,1}), \max_{j \in \{1,\ldots p\}} dbf_{i,j}^{MP,2}(t, D_{i,1})\right) \tag{13}$$

Proof: This follows from the above discussions when deriving Eq. (11) and Eq. (12). ■

With the DBF defined in Eq. (13), the same approach as at the end of Sec. IV-A by using SEIFDA can be applied. Note, that in Table II for pattern oblivious multiple paths (MP) the value of $D_{i,2}$ differs due to the different suspension intervals of τ_i^1, τ_i^2, and τ_i^3. This leads to a tighter DBF as the jump to 7 happens at $t = 15$ instead of $t = 14$ as shown in Figure 3.

C. Pattern-Clairvoyant

We assume that a task τ_i is described by a set of p triples $\{(C_{i,1}^1, S_i^1, C_{i,2}^1), \ldots, (C_{i,1}^p, S_i^p, C_{i,2}^p)\}$ each describing a possible execution pattern. We first present the corresponding demand bound functions when $D_{i,1}^j$ and $D_{i,2}^j$ with $D_{i,1}^j + S_i^j + D_{i,2}^j = T_i$ are already assigned for $j = 1, 2, \ldots, p$

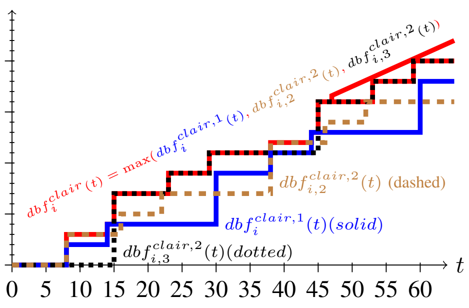

Fig. 4: SSSD: $dbf_i^{clair}(t)$ for τ_i in Table II, approximation for $g = 1$ (dotted line) after $t = g \cdot T_i + D_{i,2}^{max} = 30 + 17 = 47$. Since $dbf_{i,1}^{clair,2}(t) \leq dbf_{i,3}^{clair,2}(t)$ $\forall t$, the curve $dbf_{i,1}^{clair,2}(t)$ is omitted, but $D_{i,2}^2$ is $D_{i,2}^{max}$ in the approximation.

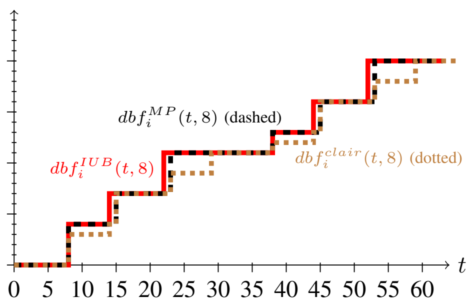

Fig. 5: Comparison of the related DBFs for the different approaches.

before considering the individual deadline assignment. For the rest of this section we implicitly assume $D_{i,1}^j + S_i^j + D_{i,2}^j = T_i$.

We define $C_{i,1}^{max}$ as $\max_{j\in\{1,\dots p\}} \left\{C_{i,1}^j\right\}$ and C_i^{max} as $\max_{j\in\{1,\dots p\}} \left\{C_{i,1}^j + C_{i,2}^j\right\}$. To calculate the workload in the period that started before t and did not finish at t, i.e., the last release before t, let $G_i^{clair}(t)$ be defined for $0 \leq t < T_i$ as:

$$G_i^{clair}(t) = \max_{j\in\{1,\dots p\}} \left\{ \begin{matrix} 0 & \text{if } t < D_{i,1}^j \\ C_{i,1}^j & \text{if } D_{i,1}^j \leq t < T_i \end{matrix} \right\} \quad (14)$$

Note, that we have to consider the maximum $C_{i,1}^j$ for each t in $0 \leq t < T_i$ as the relative deadline for the first segment of τ_i^j depends on the concrete execution/suspension pattern and is independent from the deadlines of other patterns for the same task. Similar to the analysis in Sec. IV-A and Sec. IV-B, we consider two general release patterns depending on the segment that is released at $t = 0$. If the first computation segment of task τ_i released after or at t_0 is from $C_{i,1}$, then the corresponding demand bound function $dbf_i^{clair,1}(t)$ is

$$dbf_i^{clair,1}(t) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i^{max} + G_i^{clair}\left(t - \left\lfloor \frac{t}{T_i} \right\rfloor T_i\right) \quad (15)$$

If the first computation segment of task τ_i released at or after t_0 is from $C_{i,2}^j$ the corresponding demand bound function is

$$dbf_{i,j}^{clair,2}(t) = \begin{cases} 0 & \text{if } 0 \leq t < D_{i,2}^j \\ C_{i,2}^j + dbf_i^{clair,1}(t - D_{i,2}^j) & \text{if } t \leq D_{i,2}^j \end{cases} \quad (16)$$

where $D_{i,2}^j$ is $T_i - S_i^j - D_{i,1}^j$.

Lemma 5. *The DBF of τ_i for the pattern-clairvoyant model an FRD assignment is as follows:*

$$dbf_i^{clair}(t) = \max\left(dbf_i^{clair,1}(t), \max_{j\in\{1,\dots p\}} dbf_{i,j}^{clair,2}(t)\right) \quad (17)$$

Proof: This follows from the above discussions for deriving Eqs. (15) and (16). ■

Until now, we assumed that $D_{i,1}^j$ and $D_{i,2}^j$ are assigned for $j = 1, 2, \dots, p$. For the rest of this subsection, we will discuss how to assign the relative deadlines. As we consider the scheduler to be clairvoyant, we could calculate FRDs for each of the patterns using SEIFDA and schedule the jobs with specific deadlines calculated for each execution/suspension pattern, as the pattern is known at the arrival time of a job. However, this leads to a combinatorial explosion if the number of execution/suspension patterns is large. Instead of trying to assign different deadlines for all the paths individually, we present the two following heuristics.

1) Shorter Segment Shorter Deadline (SSSD): We search for a constant relative deadline $0 < D_i^{short} \leq (T_i - S_i)/2$ assigned for the shorter computation segments of all the possible paths. This means, if $C_{i,1}^j \leq C_{i,2}^j$, then $D_{i,1}^j$ is set to D_i^{short} and $D_{i,2}^j$ is set to $T_i - S_i^j - D_i^{short}$. If $C_{i,1}^j > C_{i,2}^j$, then $D_{i,2}^j$ is set to D_i^{short} and $D_{i,1}^j$ is set to $T_i - S_i^j - D_i^{short}$.

Finding a proper D_i^{short} can be achieved by using SEIFDA directly after ordering the tasks according to $T_i - \max_{j\in\{1,\dots p\}} \left\{S_i^j\right\}$ in increasing order.

Figure 4 presents the demand bound function of task τ_i in Table II under SSSD. Note in Table II that D_i^{short} is always assigned to the smaller computation segment and that the other deadline depends on S_i^j. This leads to 3 different $dbf_{i,j}^{clair,2}(t)$. In Figure 4 only $dbf_{i,1}^{clair,2}(t)$ and $dbf_{i,3}^{clair,2}(t)$ are shown as $dbf_{i,1}^{clair,2}(t) \leq dbf_{i,3}^{clair,2}(t)$ $\forall t$.

2) Proportional Deadline with A Bias (PDAB): We intend to assign the relative deadlines proportionally to the required execution time. To avoid an arbitrarily short computation segment from being assigned with an arbitrarily short relative deadline, we introduce a constant bias D_i^{bias} for the shorter computation segments. Moreover, the relative deadline of the shorter computation segment of the j-th execution path of task τ_i should be no more than $(T_i - S_i^j)/2$.

Therefore, if $C_{i,1}^j \leq C_{i,2}^j$, the relative deadline $D_{i,1}^j$ is set to $\min\left\{(T_i - S_i^j)/2, D_i^{bias} + (T_i - S_i^j)\frac{C_{i,1}^j}{C_{i,1}^j + C_{i,2}^j}\right\}$ and $D_{i,2}^j$ is set to $T_i - S_i^j - D_{i,1}^j$. If $C_{i,1}^j > C_{i,2}^j$, then $D_{i,2}^j$ is set to $\min\left\{(T_i - S_i^j)/2, D_i^{bias} + (T_i - S_i^j)\frac{C_{i,2}^j}{C_{i,1}^j + C_{i,2}^j}\right\}$ and $D_{i,1}^j$ is set to $T_i - S_i^j - D_{i,2}^j$. For the example in Table II the bias is set to 2. Note that the case where $D_{i,1}$ is set to $(T_i - S_i^j)/2$ due to a too large bias increase happens for pattern 2 in Table II. Finding a proper D_i^{bias} can be achieved by using SEIFDA directly after ordering the tasks increasingly according to $T_i - \max_{j\in\{1,\dots p\}} \left\{S_i^j\right\}$.

D. Schedulability Tests and Approximations

Following the same argument as in the proof of the schedulability test in Eq. (5), we can replace $dbf_i^{FRD}(t, D_{i,1})$ by

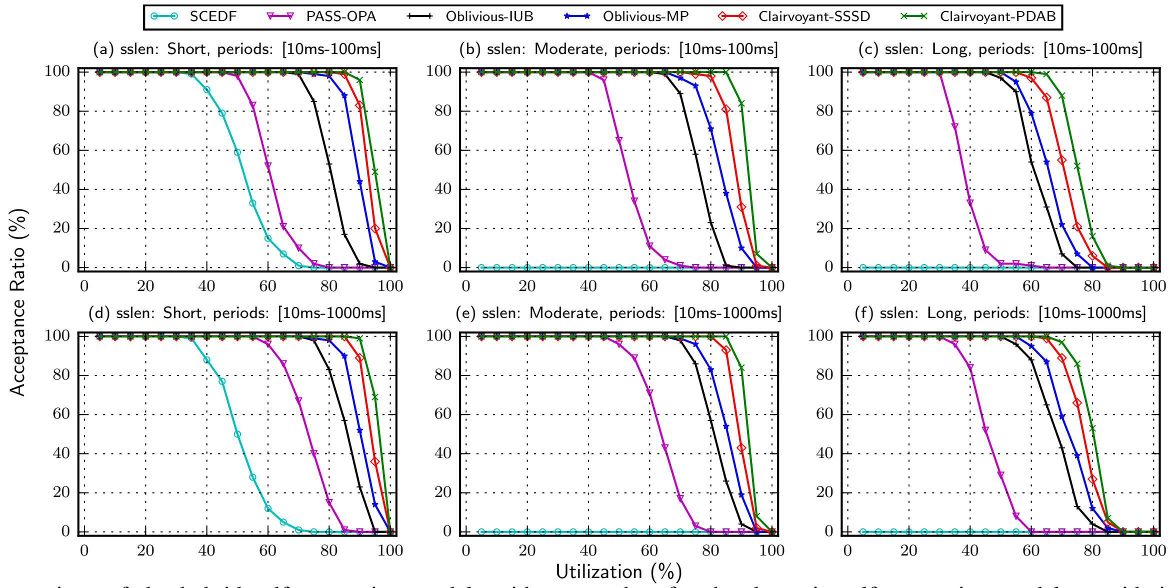

Fig. 6: Comparison of the hybrid self-suspension models with approaches for the dynamic self-suspension model, considering different suspension length (sslen) and periods in $[10ms - 100ms]$ ((a)-(c)) or $[10ms - 1000ms]$ ((d)-(f)). Approximated DBFs with $g = 2$ are used.

$dbf_i^I(t, D_{i,1})$ from Eq. (10), by $dbf_i^{MP}(t, D_{i,1})$ from Eq. (13), or by $dbf_i^{clair}(t)$ from Eq. (17), depending on the adopted hybrid self-suspension model.

Theorem 2. *An FRD schedule under a deadline assignment policy $\mathcal{A}$ is feasible if*

$$\sum_{\tau_i \in \mathbf{T}} dbf_i^{\mathcal{A}}(t, D_{i,1}) \leq t, \qquad \forall t \geq 0, \tag{18}$$

where $dbf_i^{\mathcal{A}}(t, D_{i,1})$ is defined by the adopted hybrid self-suspension strategy. That is, $dbf_i^{\mathcal{A}}(t, D_{i,1})$ can be either $dbf_i^I(t, D_{i,1})$ from Eq. (10), $dbf_i^{MP}(t, D_{i,1})$ from Eq. (13), or $dbf_i^{clair}(t)$ from Eq. (17),

Proof: This comes from Lemmas 1, 3, 4, and 5. ■

However, using the DBFs from Eq. (10), Eq. (13), or Eq. (17) directly will lead to a combinatorial explosion, as each DBF has multiple jump-points in each period and the schedulability test would have to be done for each of these jump points of each task. This problem can be tackled by using approximated DBFs. To bound the loss of the approximation, for the first g releases after t_0 the exact DBF is used and a safe linear approximation is taken afterwards, as already presented in [6], [21]. The linear approximation is taken from $t = g \cdot T_i + D_{i,2}^{max}$ and the slope is given by the task utilization U_i. To get a safe upper bound, the maximum of lines with slope U_i through all jump points in the next period is taken as linear approximation. Examples of this approximation are shown in Figure 2 and Figure 4 by the red line. This leads to a $1 + \frac{1}{g}$ approximation of the DBFs with a proof similar to the one presented in [21], that is omitted due to space limitation.

E. Comparison of Demand Bound Functions

As the different hybrid self-suspension models have access to a different amount of information, the related demand bound functions become tighter the more information can be used as shown Figure 5. $dbf_i^{clair}(t, 8) \leq dbf_i^{MP}(t, 8) \leq dbf_i^{IUB}(t, 8)$, as the clairvoyant approach can use more information than MP which can use more information than IUB.

V. Simulation Results

We conducted simulations using synthesized task sets to evaluate the proposed approaches compared with other approaches based on the *acceptance ratio* (in percent) with respect to the task set utilization. 100 task sets with a cardinality of 10 tasks were generated for each utilization level in a range from 5% to 100% with steps of 5%.

Each task set had a cardinality of 10 and we adopted the UUniFast method [2] to generated set with a given total utilization. The task periods were in *log-uniform distribution*, as suggested by Emberson et al. [10], with a period range of 10ms-100ms or 10ms-1000ms, i.e, one or two orders of magnitude, respectively. C_i and D_i where set accordingly, i.e., $C_i = T_i U_i$ and $D_i = T_i$ (implicit deadlines). We converted them to hybrid self-suspending tasks where the suspension lengths (sslen) of the tasks were generated according to a uniform distribution, in one of three ranges:

- short suspension: $[0.01(T_i - C_i), 0.1(T_i - C_i)]$
- moderate suspension: $[0.1(T_i - C_i), 0.3(T_i - C_i)]$
- long suspension: $[0.3(T_i - C_i), 0.6(T_i - C_i)]$

Each self-suspension task consisted of two paths:

- We randomly chose a path to have the largest execution time equal to C_i. The worst-case execution time of the remaining path was adjusted by multiplying with a uniformly-distributed random variable in $[0.8, 1]$.
- We randomly chose a path to have the largest suspension time equal to S_i. The worst-case suspension time of the remaining path was adjusted by multiplying with a uniformly-distributed random variable in $[0.8, 1]$.
- We then generated $C_{i,1}$ as a percentage of C_i, according to a uniform distribution, and set $C_{i,2}$ accordingly.

Note that we consider a discrete time model in the evaluation. Therefore, all task parameters were rounded up to integers. We evaluated the following approaches:

- *SCEDF*: the suspension-oblivious approach by converting suspension time into computation time.

- *PASS-OPT*: The approach for fixed-priority scheduling presented in [11]. Each interfering job is considered by running the path with the maximum cumulative execution time, i.e., C_i^{max}. Each task analyzed is considered as the task running through the path with the maximum cumulative computation and suspension time, i.e., $max_{1 \leq j \leq p}\{C_{i,1}^j + S_i^j + C_{i,2}^j\}$.
- *Oblivious-IUB*: The approach in Sec. IV-A.
- *Oblivious-MP*: The approach in Sec. IV-B.
- *Clairvoyant-SSSD*: The approach in Sec. IV-C1.
- *Clairvoyant-PDAB*: The approach in Sec. IV-C2.

The DBFs were approximated with $g = 2$ in all calculations. For *Oblivious-IUB* and *Oblivious-MP*, SEIFDA-PBminD was used, as SEIFDA-PBminD usually is the best deadline assignment strategy according to the experimental results presented in [21]. For *Clairvoyant-SSSD* and *Clairvoyant-PDAB*, we used SEIFDA-minD as a proportional lower bound is already part of the assignment in *Clairvoyant-PDAB*.

For periods in $[10ms, 100ms]$ (Figure 6(a)-(c)) we observe that the presented approaches achieve a way better acceptance ratio than state-of-the-art scheduling strategies for the dynamic self-suspension task model. The more information about the task system is used, the better the acceptance ratio gets, i.e., *Clairvoyant* approaches use more information than *Oblivious-MP* which uses more information than *Oblivious-IUB*. For the *Clairvoyant* case, *Clairvoyant-PDAB* is nearly always better than *Clairvoyant-SSSD*. While the acceptance ratio is a higher for periods in $[10ms, 1000ms]$ (Figure 6(d)-(f)) the results in general are similar and therefore further discussion omitted.

The evaluation shows that carefully using the information of the execution/suspension patterns as much as possible, in both the self-suspension model and the scheduling algorithms, will give a significant advantage with regards to schedulability. Thus, instead of focusing only the segmented and dynamic self-suspension model, the presented models and scheduling strategies should be used if possible.

VI. Conclusion

We have carefully examined a special case of the proposed hybrid self-suspension task models, in which the jobs in the system suspend themselves at most once. Depending on the knowledge of the execution/suspension patterns, we design pattern-oblivious approaches (that use the information of the patterns *offline but not online*) and pattern-clairvoyant approaches (using the information *both offline and online*). We explain how to design FRD scheduling strategies to utilize the offline patterns and develop different scheduling strategies (i.e., deadline assignments), depending on the hybrid self-suspension task models. Empirically, we show that our developed approaches are effective in terms of the acceptance ratio compared to the state-of-the-art scheduling strategies that assume the dynamic self-suspension task model.

To the best of our knowledge, this is the first result for a hybrid self-suspension task model. Although we do not explore systems with multiple self-suspension intervals in this paper, we believe that similar approaches can also be developed for the general cases when $m_i > 1$ for some tasks τ_i, as FRD was shown to be effective in this case by Huang and Chen [12].

References

[1] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17:5–22, 1999.

[2] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[3] K. Bletsas. *Worst-case and Best-case Timing Analysis for Real-time Embedded Systems with Limited Parallelism*. PhD thesis, Dept of Computer Science, University of York, 2007.

[4] B. Brandenburg. Improved analysis and evaluation of real-time semaphore protocols for P-FP scheduling. In *RTAS*, 2013.

[5] J.-J. Chen. Computational complexity and speedup factors analyses for self-suspending tasks. In *Real-Time Systems Symposium (RTSS)*, pages 327–338, 2016.

[6] J.-J. Chen and C. Liu. Fixed-relative-deadline scheduling of hard real-time tasks with self-suspensions. In *Real-Time Systems Symposium (RTSS)*, pages 149–160, 2014.

[7] J.-J. Chen, G. Nelissen, and W.-H. Huang. A unifying response time analysis framework for dynamic self-suspending tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 61–71, 2016.

[8] J.-J. Chen, G. Nelissen, W.-H. Huang, M. Yang, B. Brandenburg, K. Bletsas, C. Liu, P. Richard, F. Ridouard, Neil, Audsley, R. Rajkumar, D. de Niz, and G. von der Brüggen. Many suspensions, many problems: A review of self-suspending tasks in real-time systems. Technical Report 854, 2nd version, Faculty of Informatik, TU Dortmund, 2017. http://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/2017-chen-techreport-854-v2.pdf.

[9] H. Chetto and M. Silly-Chetto. Scheduling periodic and sporadic tasks in a real-time system. *Inf. Process. Lett.*, 30(4):177–184, 1989.

[10] P. Emberson, R. Stafford, and R. I. Davis. Techniques for the synthesis of multiprocessor tasksets. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pages 6–11, 2010.

[11] W. Huang, J. Chen, H. Zhou, and C. Liu. PASS: priority assignment of real-time tasks with dynamic suspending behavior under fixed-priority scheduling. In *Design Automation Conference*, pages 154:1–154:6, 2015.

[12] W.-H. Huang and J.-J. Chen. Self-suspension real-time tasks under fixed-relative-deadline fixed-priority scheduling. In *Design, Automation, and Test in Europe (DATE)*, pages 1078–1083, 2016.

[13] W.-H. Huang, J.-J. Chen, and J. Reineke. MIRROR: symmetric timing analysis for real-time tasks on multicore platforms with shared resources. In *Design Automation Conference (DAC)*, pages 158:1–158:6, 2016.

[14] W. Kang, S. Son, J. Stankovic, and M. Amirijoo. I/O-Aware Deadline Miss Ratio Management in Real-Time Embedded Databases. In *Real-Time Systems Symp.*, pages 277–287, 2007.

[15] C. Liu and J. Chen. Bursty-interference analysis techniques for analyzing complex real-time task models. In *Real-Time Systems Symposium (RTSS)*, pages 173–183, 2014.

[16] W. Liu, J.-J. Chen, A. Toma, T. Kuo, and Q. Deng. Computation offloading by using timing unreliable components in real-time systems. In *Design Automation Conference (DAC)*, 2014.

[17] G. Nelissen, J. Fonseca, G. Raravi, and V. Nélis. Timing Analysis of Fixed Priority Self-Suspending Sporadic Tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 80–89, 2015.

[18] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Real-Time Systems Symposium (RTSS)*, pages 26–37, 1998.

[19] B. Peng and N. Fisher. Parameter adaptation for generalized multiframe tasks and applications to self-suspending tasks. In *International Conference on Real-Time Computing Systems and Applications (RTCSA)*, pages 49–58, 2016.

[20] F. Ridouard, P. Richard, and F. Cottet. Negative results for scheduling independent hard real-time tasks with self-suspensions. In *RTSS*, pages 47–56, 2004.

[21] G. von der Brüggen, W.-H. Huang, J.-J. Chen, and C. Liu. Uniprocessor scheduling strategies for self-suspending task systems. In *International Conference on Real-Time Networks and Systems (RTNS)*, pages 119–128, 2016.