Received November 26, 2021, accepted January 13, 2022, date of publication January 14, 2022, date of current version February 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143714

Implications of Non-Uniform Deadline Scaling to Quality of Service Under Single Errors

ROBERT SCHMIDT, (Graduate Student Member, IEEE), AND ALBERTO GARCÍA-ORTIZ, (Member, IEEE)
Institute of Electrodynamics and Microelectronics, University of Bremen, 28359 Bremen, Germany
Corresponding author: Robert Schmidt (rschmidt@uni-bremen.de)

ABSTRACT Fault-tolerant real-time systems for emerging critical applications like wearable electronic healthcare monitors, consumer-grade unmanned aerial vehicles, or environmental monitoring have to tolerate errors during operation. If they fail, the consequences are dire. But often their budgets for error mitigation capabilities are low, which requires to limit mitigation capabilities to the most critical functions of a system. Still, less critical functions of a system should operate as long as possible, but current state-of-the-art scheduling approaches either ignore them or suffer from low acceptance rates. Our fully static and verification friendly mixed-criticality approach guarantees mentioned systems that the most critical system functions are always available, and maximizes the time where less critical system functions are operational: We prove that our approach is feasible, and extends the system operation time while providing full service by a factor of 1.93 with a probability of 0.92. With 1.56 higher acceptance rates compared to similar state-of-the-art approaches, the integration of functionalities of different criticality in one fault-tolerant system succeeds more and more often, which especially benefits emerging critical applications with limited budgets for error mitigation.

INDEX TERMS Real-time systems, scheduling algorithms, system recovery.

I. INTRODUCTION

To prevent catastrophic consequences, computer systems in critical applications have to tolerate faults. Accordingly, faults have to be considered during the design of such systems. A major consideration in real-time systems are faults that result in timing related errors. Their mitigation is of paramount importance to guarantee that all computations finish in time, prior to their deadline. But error mitigation capabilities are costly, as they require additional resources to provide the necessary redundancy. Therefore, it is beneficial to limit error mitigation to the highly critical functions of a system, and to provide lesser guarantees for functions of lower criticality.

In hard real-time systems, the scheduler is responsible to build a valid schedule such that all jobs finish before their deadline. In most applications, the schedulers are non-clairvoyant, and do not know a job's execution- or arrival time beforehand. To still guarantee a valid schedule, they need guarantees of worst-case execution times [1] and arrival times. These guarantees are captured in a task model. Mixed-criticality task models are the most relatable models for fault-tolerant hard real-time systems, as they allow to specify multiple estimates and therefore different levels of assurance for a job's execution time [2]. A task's criticality influences if and how a job can access the processor, and allows us to formulate different guarantees for tasks of different criticality.

Arguably the most popular mixed-criticality model, developed by Vestal [3], differentiates between low- and high criticality tasks, but guarantees execution before their deadline only for jobs from tasks of high criticality. Jobs from low criticality tasks can execute as long as the jobs from high criticality tasks did not exceed their execution time budgets, else they are discarded. This lowers the quality of service for low criticality tasks in applications where we expect execution time budget overruns.

But lowering the quality of service is not always reasonable [2], [4], which requires consideration especially for modern fault-tolerant hard real-time systems. Instead, we can strive to provide continuous degraded service, or a graceful degradation, for the low criticality tasks. The provided degraded service is based on time redundancy sourced from

The associate editor coordinating the review of this manuscript and approving it for publication was Joanna Kolodziej.

spare processing time, or slack, which is the remaining time after computations of high criticality have been served. Typically, the slack is insufficient to execute all computations from all low criticality functionalities with their desired arrival rate. Therefore, current approaches match the demand from low criticality functionalities to available slack. This can be achieved by assigning low criticality tasks a smaller execution time budget on higher levels [5], [6], or by variable [7] or scaled [8] periods. But selection of smaller execution time budgets is not possible if the budget is already at the minimum time required for the task, and partial execution provides no benefit. Moreover, variable or scaled periods require a different task model, which assumes that low criticality tasks can provide their functionality with varied job frequency. Depending on the application, this assumption is not always valid: For example, in control applications, shorter sampling periods can decrease control performance and system stability [9]. Further possibilities are to dynamically adapt tasks independent of each other [10], schedule only a subset of tasks in high criticality mode [11], or periodical dropping of low criticality jobs in high criticality mode [12]. While these approaches generate slack, they require more computational resources, due to their elaborate schedulers, compared to simpler mode-switched earliest deadline first (EDF) schedulers. Moreover, the lack of resources in deeply embedded systems rules out dynamic slack monitoring approaches [13], or to precompute proper schedules for all scenarios at design time [14]. Therefore, approaches which emphasize static design time decisions over elaborate dynamic decisions during system operation are preferable, but verification-friendly state-of-the-art approaches, which resort to static slack analysis during design time [15], [16], suffer from low acceptance rates. Hence, there is a strong need for a mixed-criticality approach for industrial fault-tolerant systems which provides degraded service to low criticality tasks, is sensible for verification, and likely finds a solution for highly loaded systems.

This work introduces an approach with these desired properties to target mentioned systems in a straightforward way: We reserve additional time by virtual deadlines to accommodate for the first error, and to guarantee service to high criticality functionalities beyond the second error.By considering the system's work done before an error, we formulate tighter worst-case bounds, which allows us to find in a wider set of systems than previous suitable virtual deadline scaling factors. Furthermore, we formulate the schedulability conditions as an optimization problem, and solve them before system deployment, resulting in a static, verification friendly approach called earliest deadline first with improved virtual deadlines for single errors (EDF-IVD-SE). We show that

- EDF-IVD-SE is feasible (Section V);
- EDF-IVD-SE is capable of tolerating a single error with affordable costs (Section VI); and
- EDF-IVD-SE doubles on average the quality of service QOS for low criticality tasks (Section X-B).

Our affordable, fault-tolerant and verification-friendly approach achieves up to 1.56 better acceptance rate compared to earliest deadline first with allowance (EDF-Allowance), while doubling the QOS for low criticality tasks on average compared to optimal traditional mixed-criticality scheduling approaches like earliest deadline first with virtual deadlines (EDF-VD).

The remainder of this work is organized as follows: After reviewing preliminaries in Section III we introduce the theory of EDF-IVD-SE in Section V. Subsequently, we show how EDF-IVD-SE can tolerate a single overrun in Section VI, and how we can formulate the search for virtual deadline scaling factors as optimization problems in Section VII. In Section VIII we describe the practical implementation of EDF-IVD-SE, followed by experimental results and conclusion in Sections X and XI.

II. NOTATION

We use the standard dual-criticality task system model, where each task τ_i in a dual-criticality sporadic task set $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ is characterized by

- a criticality $\chi_i \in \{L, H\}$;
- WCET parameters in both criticality modes c_i^L, c_i^H;
- a relative deadline d_i of the jobs of τ_i; and
- the minimum interarrival time, or period p_i between two jobs of τ_i.

For our considered implicit deadline dual-criticality task sets with independent, sporadic tasks WCET parameters, relative deadlines, and periods are related as follows:

$$\forall \tau_i \in \mathbb{T} : c_i^L \leq c_i^H \leq d_i = p_i \tag{1}$$

Each task τ_i generates an unbounded sequence of jobs. A job arrives at α_{ij} and requires γ_{ij} execution time. Job arrivals of sporadic tasks are separated at least by the task's period; in the worst case, every period a job is released. Given the WCET parameter and period, we can define the task utilization as $u_i = c_i/p_i$.

Jobs need to finish execution prior to their absolute deadlines $D_{ij} = \alpha_{ij} + d_i$. The execution time of jobs from high criticality tasks can exceed c_i^L, but never c_i^H. Low criticality jobs are not allowed to execute longer than c_i^L. If every job of high criticality tasks can execute for γ_{ij} during $[\alpha_{ij}, D_{ij})$ the task system is mixed-criticality schedulable [17].

To guarantee that every job of high criticality tasks meets its deadline, classic dual-criticality schedulers separate the system operation in two modes: low- and high criticality mode. As long as no job from a high criticality task executes longer than c_i^L, the system stays in low criticality mode, and deadlines of jobs from low- and high criticality tasks are met. If a job from a high criticality task exceeds c_i^L without signaling completion, the scheduler switches to high criticality mode. In high criticality mode, the scheduler immediately stops the release of jobs from low criticality tasks, and drops all unfinished low criticality jobs. This allows to schedule for the common case according to the optimistic WCET parameters c_i^L, while guaranteeing correctness in the uncommon case according to the pessimistic WCET parameters c_i^H.

We can express the later using task utilizations by defining utilizations U for low- and high criticality tasks and their relation to the uniprocessor's supply of $\sigma = 1$ in both modes:

$$U_L^L = \sum_{i:\chi_i=L} c_i^L/p_i \tag{2}$$

$$U_H^L = \sum_{i:\chi_i=H} c_i^L/p_i \quad U_H^H = \sum_{i:\chi_i=H} c_i^H/p_i \tag{3}$$

$$U_L^L + U_H^L \leq \sigma \quad U_H^H \leq \sigma \tag{4}$$

For the reader's convenience we provide a list of symbols and abbreviations in Section XI to summarize our notation.

III. PRELIMINARIES

This section presents two state-of-the-art approaches closely related to EDF-IVD-SE: Earliest deadline first with non-uniform virtual deadlines (EDF-NUVD) provides high chances to find schedulable solutions, but does not consider errors during operation, while EDF-ALLOWANCE accounts for errors, but only with limited chances to find schedulable solutions.

A. EDF-NUVD

Given a task set, mixed-criticality scheduling can increase the chance to find a valid schedule. We differentiate EDF-based mixed-criticality approaches by how they construct their virtual deadline scaling factors, and if they result in uniform- or non-uniform virtual deadline scaling factors. We reproduce the original EDF-NUVD schedulability proposition [17] here for the reader, as our approach in Section V is closely related.

Proposition 1 (EDF-NUVD Schedulability [17]): Let τ be a [...][dual-criticality] task [...][set] and let $0 < x_i < 1$, for each [...][high criticality task]. If

$$U_L^L + \sum_{i:\chi_i=H} u_i^L/x_i \leq 1 \tag{5}$$

$$\sum_{i:\chi_i=H} u_i^H/(1-x_i) \leq 1 \tag{6}$$

then τ is schedulable by EDF-NUVD.

Proof: See Section XI-B. □

In essence, by virtual relative deadlines we can reserve time for execution in high criticality mode, but we need to find a reservation which is still schedulable. If we choose very small virtual deadline scaling factors, we get earlier virtual relative deadlines and therefore higher task utilizations in low criticality mode according to Eq. (5), and minimal task utilizations in high criticality mode under worst case assumptions according to Eq. (6). Very large virtual deadline scaling factors result in nearly no reservation in low criticality mode, and maximum task utilizations in high criticality mode under worst case assumptions.

B. EDF-ALLOWANCE

EDF-Allowance is a EDF [18] scheduling approach which can tolerate up to k execution time budget overruns within a sliding time window W without deadline violations [15]. Similar to our EDF-IVD-SE, EDF-ALLOWANCE leverages static slack to account for overruns. For each task, the allowance is the time beyond a job's overrun which can be tolerated without compromising the deadline of any job.

Proposition 2 [15]: With a fault model k/W, which allows up to k jobs to exceed their worst-case execution time (WCET) over sliding window $W \leq \min_i p_i$, a set of sporadic tasks $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ indexed by increasing period and scheduled with EDF, a set $sp(i,k)$ of at most $k-1$ tasks but τ_i having the smallest periods in $\mathbb{T}$, and $l_i(k,t)$ as the set of $k-1$ tasks, excluding τ_i, with the highest value of $1+\lfloor (t-d_j)/p_j \rfloor$, the maximum allowance A_i^k for any faulty task τ_i is the minimum value of A_i^k satisfying the following equation:

$$\begin{aligned}
P &= lcm_i p_i \\
U^* &= \sum_{\tau_j \in \tau_i \cup sp(i,k)} \frac{A_i^k}{p_j} \\
U + U^* &\leq 1 \\
X_{i,k-1}(t) &= \sum_{d_j \leq t \wedge \tau_j \in l_i(k,t)} \left(1 + \lfloor \frac{t-d_j}{p_j} \rfloor\right) \\
t &\geq h(t) + \left(1 + \lfloor \frac{t-d_i}{p_i} \rfloor\right) A_i^k + X_{i,k-1}(t) A_i^k \\
&\forall t \in \mathcal{S} = \cup \{kp_i + d_i, k \in \mathbb{N}\} \\
&\cap \{l | \forall l \in [d_i, U^* P)\}
\end{aligned}$$

Proof: See [15]. □

The intuition behind Proposition 2 is best explained with $h(t)$, which describes the work at each point in time if each task releases jobs with its period synchronously, and is known as demand bound function DBF [19]. To derive $h(t)$ we can sum up the product of maximum computation c_i and the maximum number of requests $r(i,t)$ until time t for all tasks: $h(t) = \sum_i c_i r(i,t)$. The maximum number of requests until t is either zero or $k+1$ when $kp_i + d_i \leq t$ is satisfiable. The largest integer satisfying the former equation is $k = \lfloor t-d_i/p_i \rfloor$. This allows to define $r(i,t) = \max\left\{0, \lfloor \frac{t-d_i}{p_i} \rfloor\right\} + 1$.

The set $\mathcal{S}$ contains all check-worthy time points where $h(t) \leq t$ needs to be evaluated. The check-worthy time points are the absolute deadlines for minimum spaced requests, which describe the worst case.

With allowance, $h(t)$ needs to get extended. The term $\left(1 + \lfloor \frac{t-d_i}{p_i} \rfloor\right)$ describes the maximum number of requests of task τ_i. The sum in $X_{i,k-1}(t)$ is a sum over the maximum number of requests of the tasks in the set $l_i(k,t)$, with the maximum number of requests in t. All requests are weighted with the allowance to contribute to $h(t)$, which contains products of computation and requests.

IV. PROBLEM OVERVIEW

We want to use a fault-tolerant, real-time computer system in critical applications. Our problem is to verify that no deadline is missed *prior* to system operation, given a known scheduler, all timing-related properties of our application, and a model of faults and errors.

In our fault- and error model we consider that jobs from tasks of high criticality are protected to tolerate faults, but the protection costs additional execution time. If this additional execution time results in exceeding a task specific execution time limit, we have an error. Sometimes we refer to errors as overruns, if we want to stress the exceeding of task specific execution time limits, from a viewpoint of our application model.

A process-based overview of our EDF-IVD-SE approach is shown in Fig. 1. Given the timing-related properties of our application model, we calculate how to account for errors during system operation by virtual, earlier deadlines. If a solution exists, we can deploy the system. During system operation, a mode-switched EDF scheduler selects which jobs are allowed to access the CPU. A detailed behavioral description of the mode-switched EDF scheduler is provided in Section VIII.

The following Sections V to VII show how to solve the initially stated problem, which is to verify that no deadline is missed considering errors during system operation.

V. EDF-IVD: REDUCING THE PESSIMISM IN EDF-NUVD

The worst case assumption in Proposition 1 is very conservative, because it assumes that in high criticality mode each high criticality task needs to execute for c_i^H. This assumptions neglects that prior to the mode change some work has already finished. By taking this prior work into account, we derive tighter bounds for EDF-NUVD in this section.

Lemma 3 (Last Idle Point): The last point in time where a job in low criticality mode did not execute is at $x_i D_{ij} - c_i^L$.

Proof: With a supply of $\sigma = 1$, the uniprocessor needs at least c_i^L time units to finish a job with execution demand of c_i^L, as shown in Fig. 2. Jobs that never executed until later than c_i^L time units prior to their virtual absolute deadline would miss their virtual absolute deadline. Because no absolute deadlines or virtual absolute deadlines are missed by EDF-NUVD and earliest deadline first with non-uniform virtual deadlines for single errors (EDF-NUVD-SE) in low criticality mode if Eq. (5) and (22) hold, the jobs can't miss their deadlines, therefore the last point in time where a job in low criticality mode did not execute is at $x_i D_{ij} - c_i^L$. □

With Lemma 3 we can derive an improved version of EDF-NUVD named earliest deadline first with improved virtual deadlines (EDF-IVD):

Proposition 4 (EDF-IVD Schedulability): Let $\mathbb{T}$ be a dual-criticality implicit deadline task set and let $0 < x_i < 1$, for each high criticality task. If

$$U_L^L + \sum_{i:\chi_i = H} u_i^L / x_i \leq 1 \tag{7}$$

$$\sum_{i:\chi_i = H} u_i^H / (1 - x_i + u_i^L) \leq 1 \tag{8}$$

with mode switch from low- to high criticality mode at the first overrun time, then $\mathbb{T}$ is schedulable by EDF-IVD.

Proof: No deadline is missed by EDF-IVD in low criticality mode if Eq. (7) holds. The first time where a job from a high criticality task exceeds its low criticality WCET parameter is at t^*. Consider a job J_{ij} of a high criticality task τ_i that is active at t^*. The job arrives at α_{ij}, and has a absolute deadline at $D_{ij} = \alpha_{ij} + d_i$. Before t^*, J_{ij} is EDF-scheduled according to its virtual absolute deadline $\hat{D}_{ij} = \alpha_{ij} + x_i d_i$.

Since J_{ij} is still active at t^*, its earliest virtual absolute deadline is at the first overrun time: $\hat{D}_{ij} \geq t^*$. Therefore the duration between the first overrun time and absolute deadline is larger or equal to the duration between absolute deadline and virtual absolute deadline, as shown in Fig. 3:

$$D_{ij} - t^* \geq D_{ij} - \hat{D}_{ij} \tag{9}$$

To derive the worst case assumption considering Lemma 3, we need to differentiate two cases: *1)* first overrun time is prior to last idle point; and *2)* first overrun time is in $[\hat{D}_{ij} - c_i^L, \hat{D}_{ij}]$. If the first case $t^* < \hat{D}_{ij} - c_i^L$ is true, then $D_{ij} - t^* > D_{ij} - (\hat{D}_{ij} - c_i^L)$. By switch to high criticality mode at t^*, the job J_{ij} of high criticality task τ_i has enough time left in this case to finish prior to its deadline.

In the second case, where t^* is in $[\hat{D}_{ij} - c_i^L, \hat{D}_{ij}]$, the job had to start executing prior to t^*, because Eq. (7) holds. Still $t^* \geq \hat{D}_{ij} - c_i^L$, and $D_{ij} - t^* \leq D_{ij} - \hat{D}_{ij} + c_i^L$. But due to the low mode guarantee Eq. (7) the job executed for $t^* - (\hat{D}_{ij} - c_i^L)$, and in the remaining time $D_{ij} - t^*$ only $c_i^H - t^* + \hat{D}_{ij} - c_i^L$ need to be worked on:

$$u_{\mathrm{wc}}\left(t^*\right) = \frac{c_i^H - \left(t^* - \left(\hat{D}_{ij} - c_i^L\right)\right)}{D_{ij} - t^*} \tag{10}$$

The derivative of Eq. (10) with respect to t^* shows that Eq. (10) has no extrema:

$$\frac{\mathrm{d}u_{\mathrm{wc}}}{\mathrm{d}t^*} = \frac{c_i^H - c_i^L}{\left(t^* - \hat{D}_{ij}\right)^2} \tag{11}$$

To identify the worst case utilization u_{wc} we look at both end points in the range $[\hat{D}_{ij} - c_i^L, \hat{D}_{ij}]$ for t^*, because Eq. (10) declines with t^* and has no extrema for tasks according to Eq. (1), as shown in Fig. 4:

$$u_{\mathrm{wc}}\left(\hat{D}_{ij} - c_i^L\right) = \frac{c_i^H}{D_{ij} - \hat{D}_{ij} + c_i^L} \tag{12}$$

$$u_{\mathrm{wc}}\left(\hat{D}_{ij}\right) = \frac{c_i^H - \left(\hat{D}_{ij} - \left(\hat{D}_{ij} - c_i^L\right)\right)}{D_{ij} - \hat{D}_{ij}} \tag{13}$$

$$= \frac{c_i^H - c_i^L}{D_{ij} - \hat{D}_{ij}} \tag{14}$$

Given the utilization for both points, we still need to show which utilization is larger. A geometric interpretation is to identify the steeper slope, as shown in Fig. 5.

Lemma 5: The utilization described by Eq. (12) is larger than the utilization described by Eq. (14):

$$u_{wc}\left(\hat{D}_{ij} - c_i^L\right) - u_{wc}\left(\hat{D}_{ij}\right) > 0$$

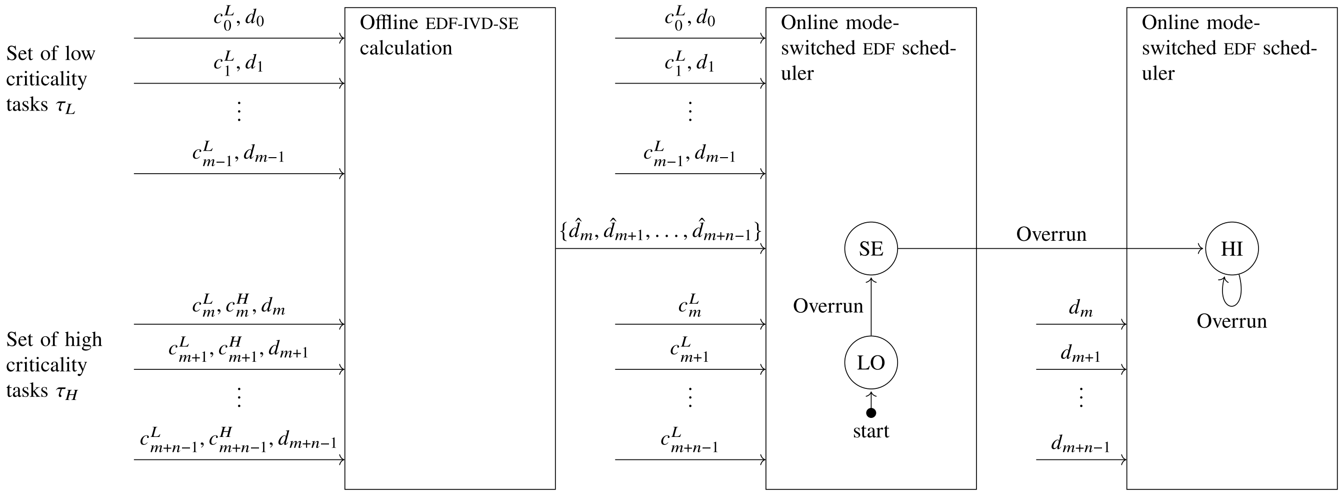

Set of low criticality tasks τ_L
c^L_0, d_0
c^L_1, d_1
c^L_{m-1}, d_{m-1}
Set of high criticality tasks τ_H
c^L_m, c^H_m, d_m
$c^L_{m+1}, c^H_{m+1}, d_{m+1}$
$c^L_{m+n-1}, c^H_{m+n-1}, d_{m+n-1}$
Offline EDF-IVD-SE calculation
$\{\hat{d}_m, \hat{d}_{m+1}, \ldots, \hat{d}_{m+n-1}\}$
c^L_m
c^L_{m+1}
c^L_{m+n-1}
Online mode-switched EDF scheduler
SE
LO
start
Overrun
Overrun
HI
Overrun
d_m
d_{m+1}
d_{m+n-1}

FIGURE 1. Overview of our approach with a task set $\mathbb{T} = \tau_L \cup \tau_H$ containing m low criticality tasks, and n high criticality tasks. Each block represents a process, which requires the information on the incoming arrows to generate the information on the outgoing arrow. The state machine diagram overlay on the right indicates when each process is active. During offline preparation, the application model is used to verify schedulability and to calculate the virtual relative deadlines $\hat{d}_i$ for all high criticality tasks by solving our optimization problem. The virtual relative deadlines are, besides schedulability verification, the key results from the offline calculations, which happen prior to online operation. During online operation, the mode-switched EDF scheduler uses the virtual relative deadlines to schedule jobs from high criticality tasks. Moreover, the system faces errors, which are considered in the application model as overruns of WCET parameters. Once the system is in high criticality mode, resembled by switching into state HI, jobs from high criticality tasks are scheduled by their original relative deadlines.

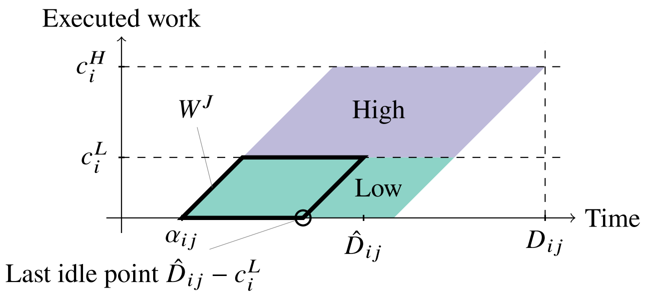


FIGURE 2. Visualization of Lemma 3 showing the job progression window W^J defined by the choice of a virtual absolute deadline $\hat{D}$. Job J_{ij} of task τ_i arrives at α_{ij}. Its fixed absolute deadline D_{ij} is $\alpha_{ij} + d_i$. Both WCET parameters are fixed as well, but the virtual absolute deadline is selected by the scheduling approach. The green shaded area below c^L_i and the purple shaded area between c^L_i and c^H_i indicate how the job can execute in low- and high criticality mode. The guarantee that in low criticality mode no deadline is missed defines the last idle point, which defines the window of possible progressions in executing a job. By selection of a virtual absolute deadline, the window can close or open fully to contain the whole green shaded area.

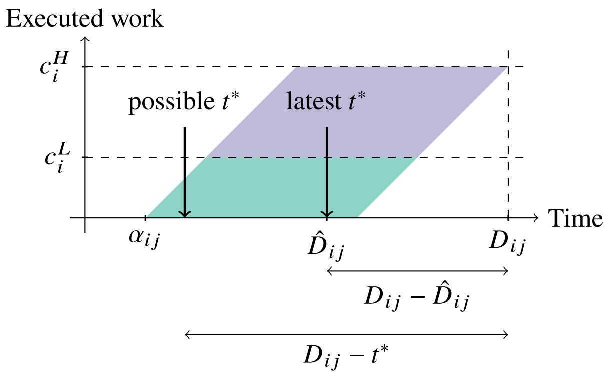


FIGURE 3. Visualization of Proposition 4. For the job to be active, the *latest* first overrun time t^* can be at the job's virtual absolute deadline $\hat{D}_i$. Whenever t^* is, the active job executed according to the shaded region below c^L_i during low criticality mode.

Proof: We show that Eq. (12) defines the worst case by contradiction, rewriting the difference as follows:

$$\frac{c^H_i}{D_{ij} - \hat{D}_{ij} + c^L_i} - \frac{c^H_i - c^L_i}{D_{ij} - \hat{D}_{ij}} \leq 0 \tag{15}$$

$$D_{ij} - \hat{D}_{ij} + \leq c^H_i - c^L_i \tag{16}$$

$$1 \leq \frac{c^H_i - c^L_i}{D_{ij} - \hat{D}_{ij}} \tag{17}$$

We can reason about the contradiction in two ways: *1)* If Eq. (16) is true, the reserved time for high criticality work can be less than actual high criticality work, contradicting schedulability; or *2)* Due to the problem definition in Eq. (1), where $c^L_i = c^H_i$ is possible, the difference in Eq. (17) can become zero, leading to a contradiction. Therefore the utilization described by Eq. (12) is greater than the utilization described by Eq. (14). □

With the worst case utilization identified by Lemma 5, we can rewrite Eq. (12) to get rid of absolute deadlines and virtual absolute deadlines:

$$u_{\mathrm{wc}} = \frac{c^H_i}{D_{ij} - \hat{D}_{ij} + c^L_i} = \frac{c^H_i}{D_{ij}\left(1 - x_i + c^L_i / D_{ij}\right)} \tag{18}$$

Replacing the job's absolute deadline D_{ij} with $\alpha_{ij} + d_i$, and considering a synchronous release pattern of jobs from all

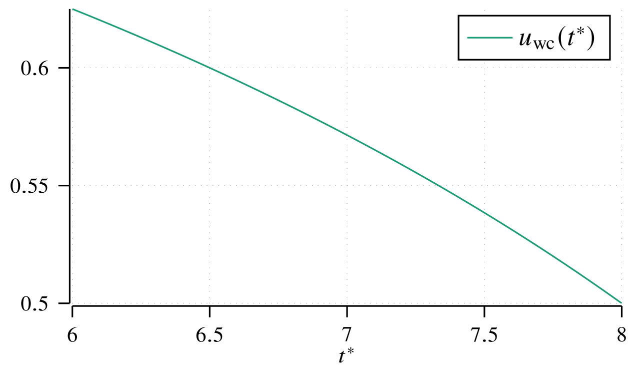
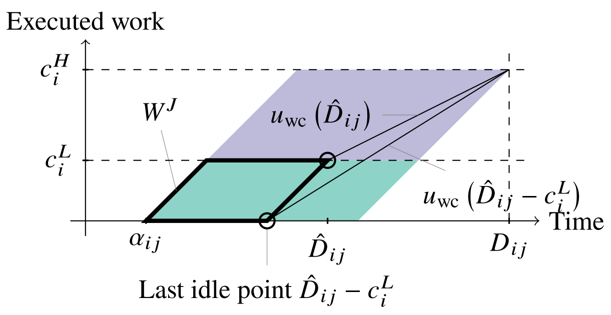


FIGURE 4. Evaluation of Eq. (10) for example task and corresponding job with $c_i^L = 2, c_i^H = 5, D_{ij} = 14, \hat{D}_{ij} = 8$.



Executed work
c_i^H
W^J
$u_{\mathrm{wc}}(\hat{D}_{ij})$
c_i^L
$u_{\mathrm{wc}}(\hat{D}_{ij} - c_i^L)$
Time
α_{ij}
$\hat{D}_{ij}$
D_{ij}
Last idle point $\hat{D}_{ij} - c_i^L$

FIGURE 5. The steepest slope is observed at the last idle point, which corresponds to the largest utilization as described by Eq. (12). Both slopes start at the boundary of the job progression window W^J. In this example, with $c_i^L = 2, c_i^H = 5, D_{ij} = 14, \hat{D}_{ij} = 8$, the slopes are $u_{\mathrm{wc}}\left(\hat{D}_{ij} - c_i^L\right) = 5/8$ and $u_{\mathrm{wc}}\left(\hat{D}_{ij}\right) = 4/8$.

tasks $\forall i \; \alpha_{i0} = 0$ yields:

$$u_{\mathrm{wc}} = \frac{c_i^H}{d_i\left(1 - x_i + c_i^L/d_i\right)} = \frac{u_i^H}{1 - x_i + u_i^L} \tag{19}$$

Now Eq. (8) regards the task set after t^* as a single-criticality task set under the worst case, where each high-criticality task τ_i has an increased task utilization as described by Eq. (19). If the sum over all increased task utilizations is below the supply of one, this single-criticality task set is schedulable. □

VI. EXTENSION TO SINGLE OVERRUN TOLERANCE

In our anticipated applications WCET parameter overruns are expected [16]. As traditional dual-criticality scheduling discards low criticality tasks upon the first overrun, under the assumption that overruns are unlikely, we propose the following extensions for single overrun tolerance to provide better service to low criticality tasks.

First, we show how EDF-NUVD can be extended to tolerate a single overrun, which we call EDF-NUVD-SE, then we extend EDF-IVD to EDF-IVD-SE which can tolerate a single overrun as well.

A. SINGLE ERROR EXTENSION OF EDF-NUVD

To derive the schedulability conditions for EDF-NUVD-SE we reserve additional time in low criticality mode to tolerate a single overrun. We increase the virtual utilization of high criticality tasks in low criticality mode $\hat{U}_H^L = \sum_{i:\chi_i=H} u_i^L/x_i$ by accounting for c_j^H instead of c_j^L:

$$\hat{U}_H^L = \sum_{i:\chi_i=H} u_i^L/x_i \tag{20}$$

$$\leq u_j^H/x_j + \sum_{\substack{i:\chi_i=H \\ i\neq j}} u_i^L/x_i = \tilde{U}_H^L \tag{21}$$

If we replace $\hat{U}_H^L$ with the increased virtual utilization $\tilde{U}_H^L$ in Eq. (5), task j can use its high criticality WCET parameter in low criticality mode. To allow any n_H high criticality tasks to overrun, we need to satisfy n_H low criticality mode equations, and the unchanged high criticality mode equation Eq. (6):

Proposition 6 (EDF-NUVD-SE schedulability): Let $\mathbb{T}$ be a dual-criticality implicit deadline task set and let $0 < x_i < 1$, for each high criticality task. If

$$\forall j \quad U_L^L + u_j^H/x_j + \sum_{\substack{i:\chi_i=H \\ i\neq j}} u_i^L/x_i \leq 1 \tag{22}$$

$$\sum_{i:\chi_i=H} u_i^H/(1 - x_i) \leq 1 \tag{23}$$

with mode switch from low- to high criticality mode at the second overrun time, then $\mathbb{T}$ is schedulable by EDF-NUVD-SE.

Proof: If Eq. (1) holds, then $u_j^H/x_j \geq u_j^L/x_j$ and therefore $\tilde{U}_H^L \geq \hat{U}_H^L$. No deadline is missed by EDF-NUVD-SE in low criticality mode until the first overrun at t^* if Eq. (22) holds. The first overrun stems from a job J_{ij} of a high criticality task τ_i that is active at t^* with $\gamma_{ij} > c_j^L$. Since we reserved $u_i^H/x_i = c_i^H/(x_i d_i)$ and $\gamma_{ij} \leq c_i^H$ for any task, no deadline is missed until the second overrun time at $t^{\circledast}$.

Consider another job J_{kl} active at $t^{\circledast}$ which arrives at α_{kl} and has a absolute deadline D_{kl} and virtual absolute deadline $\hat{D}_{kl}$. Prior to $t^{\circledast}$, J_{kl} is EDF-scheduled according to its virtual absolute deadline. Since J_{kl} is active at $t^{\circledast}$ the virtual absolute deadline is at or later than the second overrun time $\hat{D}_{kl} \geq t^{\circledast}$. Therefore the duration between absolute deadline and virtual absolute deadline is the minimum duration between absolute deadline and second overrun time:

$$D_{kl} - t^{\circledast} \geq D_{kl} - \hat{D}_{kl} \tag{24}$$

In the worst case each high criticality task has an active job with execution time equal to WCET parameter in high criticality mode, and the execution needs to takes place during $D_{kl} - \hat{D}_{kl}$. Regarding the task set after the second overrun time as a single-criticality task set under the worst case assumption, results in Eq. (23). If the worst case utilization is below the supply of one, this single-criticality task set is schedulable. Hence, EDF-NUVD-SE can schedule $\mathbb{T}$. □

B. SINGLE ERROR EXTENSION OF EDF-IVD

Identical to EDF-NUVD-SE we reserve additional time in low criticality mode to tolerate a single overrun. We increase the virtual utilization of high criticality tasks in low criticality mode from $\hat{U}_H^L$ to $\tilde{U}_H^L$ by accounting for c_j^H instead of c_j^L as in Eq. (21).

By replacing $\hat{U}_H^L$ with $\tilde{U}_H^L$ in Eq. (7), task j can use its high criticality WCET parameter in low criticality mode. Because it is unknown which task overruns, we need to consider that any of n_H high criticality tasks can overrun. This increases the number of low criticality mode equations to n_H:

Proposition 7 (EDF-IVD-SE schedulability): Let $\mathbb{T}$ *be a dual-criticality implicit deadline task set according to Eq.* (1), *and let* $0 < x_i < 1$, *for each high criticality task. If*

$$\forall j \quad U_L^L + u_j^H/x_j + \sum_{\substack{i:\chi_i=H\\ i\neq j}} u_i^L/x_i \leq 1 \tag{25}$$

$$\sum_{i:\chi_i=H} u_i^H/(1-x_i+u_i^L) \leq 1 \tag{26}$$

with mode switch from low- to high criticality mode at the second overrun time, then $\mathbb{T}$ *is schedulable by EDF-IVD-SE.*

Proof: As $u_j^H/x_j \geq u_j^L/x_j$ the increased virtual utilization of high criticality tasks in low criticality mode is larger or equal than their virtual utilization $\tilde{U}_H^L \geq \hat{U}_H^L$. If Eq. (25) holds no deadline is missed by EDF-IVD-SE in low criticality mode until the first overrun at t^*. The first overrunning job J_{ij} is from a high criticality task τ_i that is active at t^* with $\gamma_{ij} > c_j^L$. Since we reserved $u_i^H/x_i = c_i^H/(x_i d_i)$ and $\gamma_{ij} \leq c_i^H$ for any task, no deadline is missed until the second overrun time at $t^{\circledast}$.

Consider another job J_{kl} active at $t^{\circledast}$ which arrives at α_{kl} and has a absolute deadline D_{kl} and virtual absolute deadline $\hat{D}_{kl}$. Prior to $t^{\circledast}$, J_{kl} is EDF-scheduled according to its virtual absolute deadline. Since J_{kl} is active at $t^{\circledast}$ the virtual absolute deadline is at or later than the second overrun time $\hat{D}_{kl} \geq t^{\circledast}$.

As in Proposition 4, the worst case according to Lemma 3 is where $t^{\circledast}$ is at $\hat{D}_{kl} - c_k^L$:

$$u_{\text{wc}} = \frac{c_k^H}{D_{kl} - \hat{D}_{kl} + c_k^L} = \frac{u_k^H}{1 - x_k + u_k^L} \tag{27}$$

Therefore Eq. (26) considers that each task in the task set has an active job after the second overrun time as described in the worst case above, which can be successfully EDF-scheduled if the resulting utilization is below the uniprocessor's supply of one. □

VII. SOLVING FOR VIRTUAL DEADLINE SCALES

In this section we solve the schedulability conditions for EDF-NUVD-SE, EDF-IVD, and EDF-IVD-SE to get a solution for the virtual deadline scaling factors. With the virtual deadline scaling factors we can calculate in advance the virtual relative deadlines for all tasks, which are required by the scheduler during operation. Moreover, we are interested in a solution for the virtual deadline scaling factors which allows the maximum amount of low criticality work.

For each approach we formulate an optimization problem where the schedulability conditions are the constraints, and the low criticality work defines the objective function to maximize. For any task set with n_H high criticality tasks the vector of variables is $\boldsymbol{y} = \left[x_0\ x_1\ \ldots\ x_{n_H-1}\ U_L^L\right]$, and the objective function is $f(\boldsymbol{y}) = U_L^L$.

For EDF-NUVD-SE the schedulability conditions Eqs. (22) and (23) are the constraints in our optimization:

Optimization problem 8 (EDF-NUVD-SE):

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{maximize}}\ U_L^L \\ &\text{subject to}\ \forall j\ 1 - U_L^L - u_j^H/x_j - \sum_{\substack{i:\chi_i=H\\ i\neq j}} u_i^L/x_i \geq 0 \\ &1 - \sum_{i:\chi_i=H} u_i^H/(1-x_i) \geq 0 \end{aligned} \tag{28}$$

In EDF-IVD the constraints are Eqs. (7) and (8). Note that EDF-IVD requires fewer constraints than EDF-NUVD-SE or EDF-IVD-SE, as the low criticality schedulability condition is the same for all tasks:

Optimization problem 9 (EDF-IVD):

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{maximize}}\ U_L^L \\ &\text{subject to}\ 1 - U_L^L - \sum_{i:\chi_i=H} u_i^L/x_i \geq 0 \\ &1 - \sum_{i:\chi_i=H} u_i^H/(1-x_i+u_i^L) \geq 0 \end{aligned} \tag{29}$$

Finally EDF-IVD-SE with its schedulability conditions Eqs. (7) and (8) results in the following optimization problem:

Optimization problem 10 (EDF-IVD-SE):

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{maximize}}\ U_L^L \\ &\text{subject to}\ \forall j\ 1 - U_L^L - u_j^H/x_j - \sum_{\substack{i:\chi_i=H\\ i\neq j}} u_i^L/x_i \geq 0 \\ &1 - \sum_{i:\chi_i=H} u_i^H/(1-x_i+u_i^L) \geq 0 \end{aligned} \tag{30}$$

Both Optimization problems 8 and 10 have $n_H + 1$ nonlinear inequality constraints and a scalar objective function. Note Optimization problem 9 has two nonlinear inequality constraints and a scalar objective function, and all optimization problems are solvable by nonlinear programming NLP.

We use sequential least squares programming SLSQP [20], [21] to solve Optimization problems 8 to 10, as all functions are twice continuously differentiable. On success, the resulting virtual deadline scaling factors and maximum utilization of low criticality tasks can be used to schedule the task set.

VIII. SYSTEM OPERATION

During system operation, our mode-switched EDF scheduler is in one of three modes, as shown in Fig. 6: *1)* initial low criticality mode; *2)* intermediate single error mode; and *3)* high criticality mode. Modes are switched as in earliest deadline first with virtual deadlines for single errors EDF-VD-SE [16]:

During the initial low criticality mode, all tasks are allowed to generate jobs. The jobs arrive at the scheduler's queue, and are EDF-scheduled according to their absolute deadlines or virtual absolute deadlines: Jobs from low criticality tasks are scheduled according to their absolute deadlines, and jobs from high criticality tasks are scheduled according to their virtual absolute deadlines. The mode- and task criticality dependent deadline selection is also shown in Fig. 7.

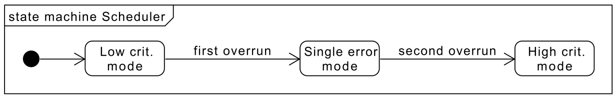


FIGURE 6. UML state machine diagram of EDF-NUVD-SE mode switching.

As soon as a job is granted access to the CPU by the scheduler, as shown in Fig. 8 it starts to execute, and the scheduler keeps track of the time since the job started. The scheduler is unknowledgeable of the time it takes for the job to actually finish, but the WCET parameter is known. If the time since the job started is larger than the WCET parameter, the scheduler kills the job if it is from a low criticality task, or switches to the intermediate single error mode if it is from a high criticality task.

During the intermediate single error mode, the scheduler operates as in the initial low criticality mode, but if a job from a high criticality task takes longer than the WCET parameter, the system is switched to high criticality mode.

In high criticality mode, no jobs from low criticality tasks are generated, as shown in Fig. 7, and jobs from high criticality tasks are scheduled according to their original deadline.

A. ERROR MODEL

Our approach targets emerging critical applications, where we expect errors, and yet the budget to deal with them is very limited compared to traditional critical applications. Such semi-critical applications, like wearable healthcare monitors, are required to be dependable, while being built from commercial off-the-shelf COTS components.

The key to dependable design with COTS components are error detection and correction. Both can be implemented in software by exploiting temporal redundancy. In our model, the time for error detection is part of the WCET parameter in low criticality mode for high criticality tasks, and we consider error correction by recomputing wrong results in the WCET of high criticality tasks.

Therefore, we model the pessimism in estimating the task's WCET for error correction with $c_i^H = z_i c_i^L$, where $z_i \in \{x \in \mathbb{Z} : x > 1\}$.

IX. CASE STUDY

To exemplify the usefulness of EDF-IVD-SE we investigate a flight management system usecase [22] shown in Table 1. While the task set is worst-case schedulable with EDF, it is not schedulable with EDF-Allowance: A worst-case reservation results in an utilization of $U = 1993/2000 < 1$, with nearly no static slack for any allowance.

```
Require: Task $\tau_i$
Ensure: Job $J_{ij}$ with mode-dependent deadline
 1: loop
 2:     if Task $\tau_i$ ready $\wedge$ task period $p_i$ passed then
 3:         $M \leftarrow$ GetCriticalityMode
 4:         if $\chi_i = H$ then
 5:             if $M = H$ then
 6:                 return Job with absolute deadline
 7:             else
 8:                 return Job with virtual absolute deadline
 9:             end if
10:         else                                  ▷ Low criticality task
11:             if $M = H$ then
12:                 return No job                 ▷ Dropped
13:             else
14:                 return Job with absolute deadline
15:             end if
16:         end if
17:     end if
18: end loop
```

FIGURE 7. Mode-dependent CPU requests (jobs) from a sporadic task. If the task is ready, and the last job is at least *p* time units ago, a new job can be created where the absolute deadline is selected based on the task's criticality and the current system mode. For jobs from tasks with a high criticality level *H*, virtual absolute deadlines are used as long as the system is *not* in high criticality mode. jobs from tasks with a low criticality level *L* use absolute deadlines, as long as the system is *not* in high criticality mode. In high criticality mode, no jobs from tasks with low criticality levels are generated.

But with EDF-IVD-SE the task set is schedulable: Solving the optimization problem, we find a solution with virtual deadline scaling factors as shown in Table 2 and maximum supported utilization in low criticality mode of ≈ 0.59. The task set's utilization in low criticality mode is 0.62, which guides us to lower the task set's utilization by ≈ 0.03. Depending on the application, it might be sensible to remove low criticality tasks, extend their period, or reduce their WCET parameter. For this example, we reduce the WCET parameter of task 9, 10 and 11, as shown in Table 3. Even the adjusted task set is not schedulable by EDF-Allowance, but successfully schedulable with EDF-IVD-SE.

TABLE 1. Flight management system usecase [22].

Task i	p_i	d_i	c_i^L	c_i^H
1	5000	5000	10	20
2	200	200	10	20
3	1000	1000	10	20
4	1600	1600	10	20
5	100	100	10	20
6	1000	1000	10	20
7	1000	1000	10	20
8	1000	1000	20	–
9	1000	1000	200	–
10	1000	1000	200	–
11	1000	1000	200	–

```
Require: Scheduling event $e$
Require: Jobs $J_{ij}$ in priority queue sorted by earliest deadline
Ensure: Access to CPU for all jobs
 1: procedure FULLSERVICE(Event $e$, Job $J^Q$, Job $J^R$)
 2:     if $e$ = new job in queue then
 3:         if deadline of $J^Q$ < deadline of $J^R$ then
 4:             Preempt and move $J^R$ to priority queue
 5:             Run job $J^Q$ on CPU
 6:         end if
 7:     else if $e$ = current job finishes then
 8:         Run job $J^Q$ on CPU
 9:     end if
10: end procedure
11: procedure REDUCEDSERVICE(Event $e$, Job $J^Q$, Job $J^R$)
12:     $\chi^Q \leftarrow$ LOOKUPCRITICALITY($J^Q$)
13:     if $e$ = new job in queue then
14:         if $\chi^Q = L$ then
15:             drop $J^Q$
16:         else if deadline of $J^Q$ < deadline of $J^R$ then
17:             Preempt and move $J^R$ to priority queue
18:             Run job $J^Q$ on CPU
19:         end if
20:     else if $e$ = current job finishes then
21:         if $\chi^Q = H$ then
22:             Run job $J^Q$ on CPU
23:         else
24:             drop $J^Q$
25:         end if
26:     end if
27: end procedure
28: loop
29:     $M \leftarrow$ GETCRITICALITYMODE
30:     $J^R \leftarrow$ Job currently running on CPU
31:     $J^Q \leftarrow$ Job from priority queue
32:     $e \leftarrow$ Next event
33:     if $e$ = detect overrun $\wedge$ $(M \neq H)$ then
34:         Increase system criticality mode
35:         continue
36:     end if
37:     if $M = L$ then                          ▷ Low criticality mode
38:         FULLSERVICE($e, J^Q, J^R$)
39:     else if $M = S$ then                     ▷ Single error mode
40:         FULLSERVICE($e, J^Q, J^R$)
41:     else                                     ▷ High criticality mode
42:         REDUCEDSERVICE($e, J^Q, J^R$)
43:     end if
44: end loop
```

FIGURE 8. Event-based description of mode-switched EDF scheduler used in EDF-IVD-SE. The scheduler operates in an endless loop, spanning line 29 to 44, to react to a stream of events *e*. Possible events are the arrival of a new job in the priority queue, the completion of the job currently running on the CPU, or the detection of a WCET parameter overrun from a job of a high criticality task. Overruns are handled first, in line 33 to 36, by increasing the criticality mode of the system if it is not yet in high criticality mode. The response to other events is mode-dependent and handled in line 37 to 43. In low- and single error mode, all jobs can access the CPU according to their deadlines. In high criticality mode, jobs from tasks of low criticality are dropped, and only remaining jobs from high criticality tasks can access the CPU.

X. EXPERIMENTS

In this section we quantify the amount of schedulable task systems with EDF-NUVD-SE, EDF-IVD-SE, and the benefit in QOS. To investigate the amount of schedulable task sets we generate random task sets and solve the corresponding optimization problem. We report the number of schedulable task sets over the total number of task sets, or *acceptance rate*, over increasing utilization in low criticality mode.

For QOS, we evaluate the additional time the system is operational after the first overrun time with `Thready` [23] by simulating a large set of random task sets.

TABLE 2. EDF-IVD-SE virtual deadline scales for task set in Table 1.

Task i	x_i
1	0.603 009 38
2	0.631 890 57
3	0.607 817 98
4	0.605 562 71
5	0.749 381 33
6	0.607 817 98
7	0.607 817 98

TABLE 3. Adjusted flight management system usecase.

Task i	p_i	d_i	c_i^L	c_i^H
1	5000	5000	10	20
2	200	200	10	20
3	1000	1000	10	20
4	1600	1600	10	20
5	100	100	10	20
6	1000	1000	10	20
7	1000	1000	10	20
8	1000	1000	20	–
9	1000	1000	190	–
10	1000	1000	190	–
11	1000	1000	190	–

A. ACCEPTANCE RATE OF `UUnifast` RANDOM TASK SYSTEMS FOR DIFFERENT UTILIZATIONS

Reporting the acceptance rate with random task sets requires to apply the schedulability check, and in case of EDF-NUVD-SE, EDF-IVD, and EDF-IVD-SE this includes solving the optimization problems presented in Section VII.

For our investigation we resort to random task sets, generated with the `UUnifast` algorithm [24], additional pessimism [16], and a mostly EDF-VD schedulable parameterization [25]: Periods are uniformly drawn between $p_l = 50$ and $p_u = 200$, and pessimism is uniformly drawn between $z_l = 1$ and $z_u = 2$. For each utilization in low criticality mode, we calculate the acceptance rate from 1024 task sets. Moreover, we compare the acceptance rates for a parameterization typically found in automotive and avionics systems with periods between $p_l = 25$ and $p_u = 1000$ [13].

In general, a higher acceptance rate is better, because it indicates a higher chance to accept a task set as schedulable. As the utilization in low criticality mode U_L increases, the difficulty to find acceptable virtual deadline scaling factors increases as well.

We compare the acceptance rate in Fig. 9 for approaches without single error tolerance. All approaches decline in acceptance rate with increasing utilization in low criticality mode. It stands out that EDF-VD dominates both EDF-NUVD and EDF-IVD despite having only a single deadline scaling parameter. By intuition, approaches with individual deadline scaling parameters should be at least as powerful as uniform deadline scales. But the proof for EDF-VD feasibility [17] results in tighter bounds for the worst case, which increases the chance to deem a task set schedulable. Nevertheless, the acceptance rate of EDF-IVD is at least as good or better than the acceptance rate of EDF-NUVD. For the avionic- and automotive-like task sets in Fig. 11 the trends are similar, although the absolute acceptance rates are lower.

Extending all three approaches to tolerate a single error drastically changes the acceptance rate as seen in Fig. 10. For utilization up to 0.65 EDF-IVD-SE provides the best acceptance rate. Contrary, the uniform deadline scale of EDF-VD-SE provides less room for the numerical solver to find a solution. Similar to the approaches without single error tolerance, the acceptance rate trends for the avionic- and automotive-like task sets in Fig. 12 are alike, while absolute acceptance rates are lower. Nevertheless, in terms of acceptance rate all approaches are sensitive to the composition of their task sets. Therefore, detailed descriptions about random task set generation [16] are a necessity for comparable results.

It is fundamental to note that the analytical solution for EDF-VD is optimal if errors are *not* considered, but with errors the analytical solution is not applicable anymore. Moreover, with just a single free parameter, the numerical solver has less opportunities to find a solution for EDF-VD-SE, compared to EDF-NUVD-SE and EDF-IVD-SE, resulting in reduced acceptance rates.

Comparing the acceptance rates of EDF-IVD to EDF-IVD-SE in Fig. 13 for the same task sets highlights the cost of tolerating a single error in terms of acceptance rate. With a maximum cost of ≈ 0.146 EDF-IVD lends itself for single error extension. In the same way we can compare the acceptance rates of EDF to EDF-Allowance. The maximum cost in terms of acceptance rate is ≈ 0.33, as shown in Fig. 14, instead of ≈ 0.146 from our approach.

Moreover, utilizations of larger than 0.55 are increasingly difficult for EDF-Allowance, and the start of a decline in acceptance rate compared to EDF-IVD-SE. With up to ≈ 1.56 better acceptance rate EDF-IVD-SE can more likely schedule highly loaded task sets.

B. QUALITY OF SERVICE COMPARISON BY MODE SWITCH TIME

Both EDF-IVD-SE and EDF-NUVD-SE are designed to tolerate a single overrun without discarding all low criticality tasks, prolonging the time when jobs from low criticality tasks can execute until the second overrun. This prolonged time results in additional service for low criticality tasks. Therefore, we define the QOS as the ratio of second overrun time to first overrun time: $t^{\circledast}/t^{*}$.

We investigate the QOS of EDF-IVD-SE and EDF-NUVD-SE by simulation with `Thready`, which supports EDF scheduling simulations of mixed-criticality task sets with fixed overrun probabilities. We select EDF-IVD-SE and EDF-NUVD-SE schedulable dual-criticality task sets which we generate with the `UUnifast` algorithm and generation parameters

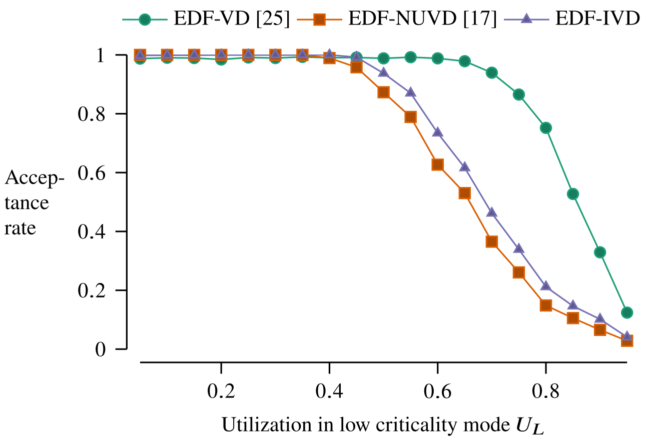


FIGURE 9. Acceptance rate for `UUnifast` random task system. A higher acceptance rate is better. With increasing utilization on low criticality mode U_L the difficulty to find acceptable deadline scales increases.

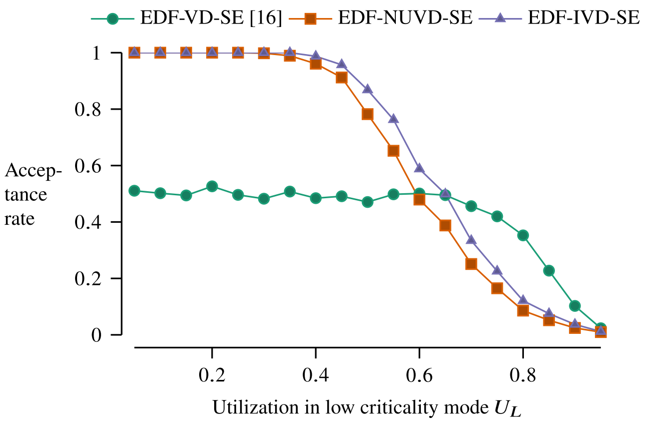


FIGURE 10. Acceptance rate of random task sets for approaches with single error tolerance.

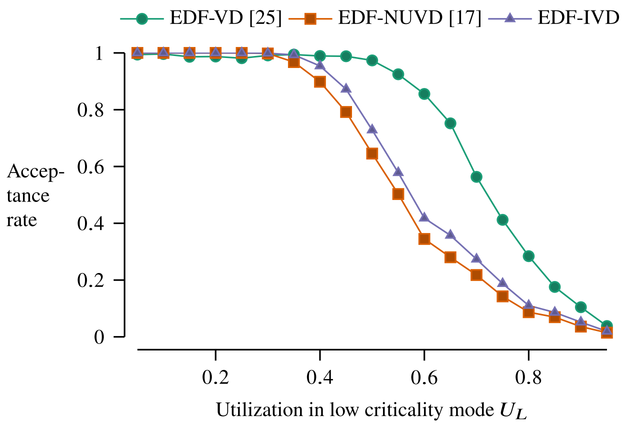


FIGURE 11. Acceptance rate of random automotive-like task sets for approaches without single error tolerance.

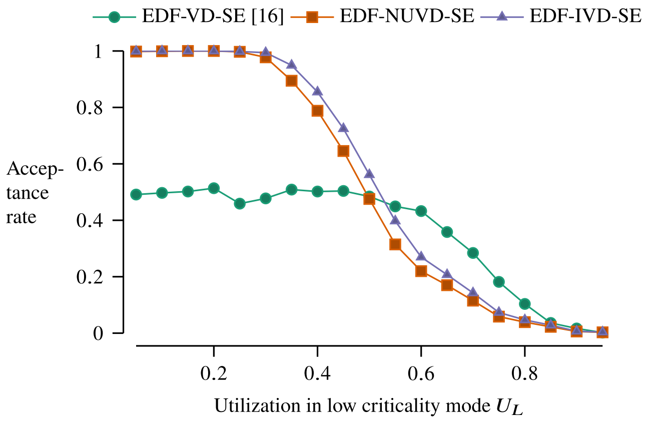


FIGURE 12. Acceptance rate of random automotive-like task sets for approaches with single error tolerance.

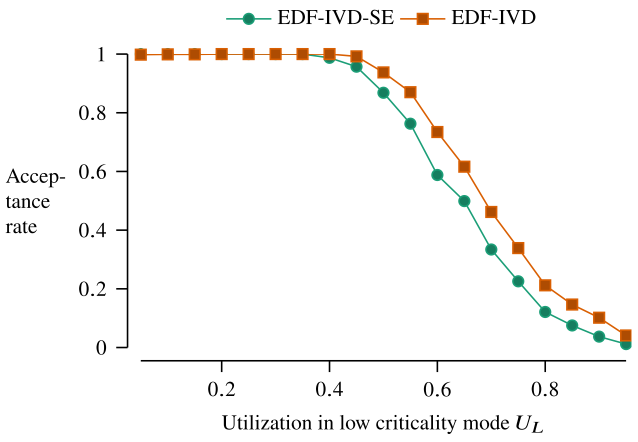


FIGURE 13. EDF-IVD-SE lends itself for single error extension. Plot shows the acceptance rate for `UUnifast` random task sets with EDF-VD-friendly parameterization [25]. The acceptance rate of EDF-IVD-SE is only slightly reduced compared to EDF-IVD. Therefore EDF-IVD-SE makes error tolerance affordable in most cases.

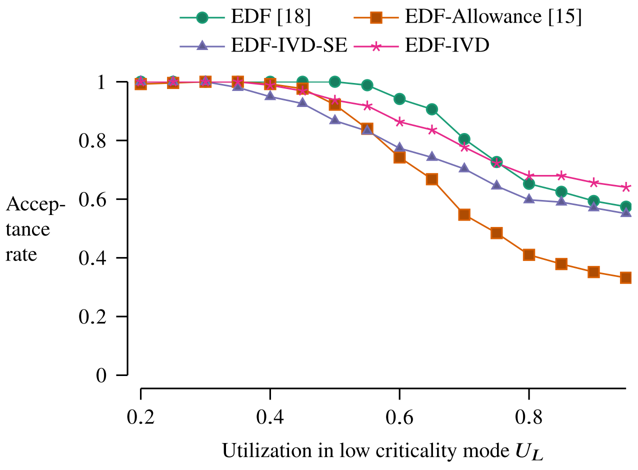


FIGURE 14. Acceptance rate of random task sets. The acceptance rate of EDF-IVD-SE is better than the allowance approach.

as described in Section X-A. The random task sets are simulated with millisecond time step resolution for one hour.

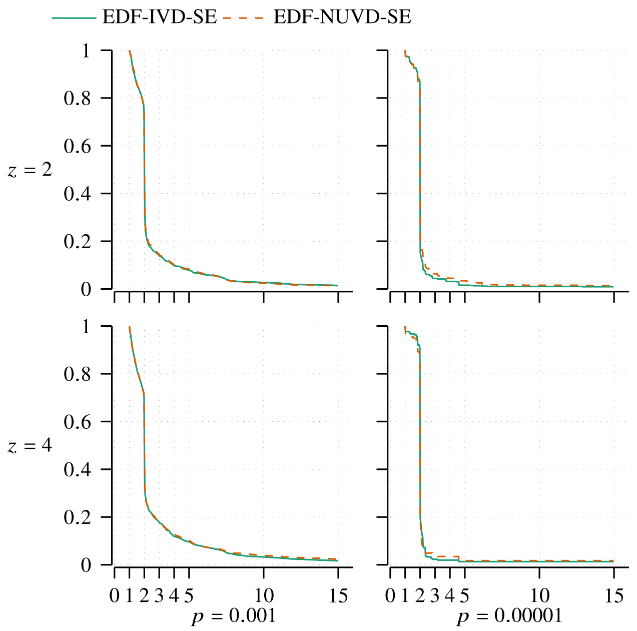


FIGURE 15. CCDF, or survival function, of EDF-IVD-SE and EDF-NUVD-SE QOS for error probabilitys $p = 0.001$ and $p = 0.0001$. Rows differentiate pessimism, and columns differentiate error probability. In each plot, the ordinate abscissa Each plot shows the probability (ordinate) of a system to survive until $t^{\circledast}/t^{*}$ (abscissa).

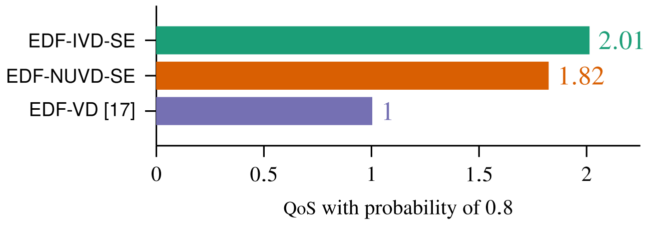


FIGURE 16. Achievable QoS for EDF-IVD-SE and EDF-NUVD-SE with probability of 0.8 compared to baseline EDF-VD. Both EDF-IVD-SE and EDF-NUVD-SE can tolerate a single error, and therefore double on average the QoS for low criticality tasks.

While all approaches can tolerate the first overrun deliberately, it is still interesting to quantify the probabilities to achieve at least a specific QOS.

In Fig. 15 with an error probability of $p = 10^{-3}$ and pessimism of two EDF-NUVD-SE achieves at least a QOS of 1.82 with a probability of 0.8, or a QOS of 2.01 with a probability of 0.46. For the same scenario, EDF-IVD-SE achieves at least a QOS of 2.01 with a probability of 0.45, or a QOS of 1.85 with a probability of 0.8. Similar results are achieved for a pessimism of four, and error probability of 10^{-5}, with a QOS of 1.93 at a probability of 0.92. Therefore both EDF-IVD-SE and EDF-NUVD-SE double on average the QoS for low criticality tasks compared to baseline EDF-VD, as visualized in Fig. 16.

From a global view, both EDF-NUVD-SE and EDF-IVD-SE provide the same overrun tolerance by design, and are similar in their capabilities beyond the first overrun. Nevertheless, EDF-IVD-SE is preferable, as it is more likely to find valid virtual deadline scaling factors for a given task set.

Furthermore, if we compare EDF-IVD-SE with EDF-Allowance, we immediately see that EDF-Allowance has no service guarantee, which results in rising likelihood of mission failures with rising utilization as shown in Fig. 17. Contrary, EDF-IVD-SE has a service guarantee, which guarantees mission success.

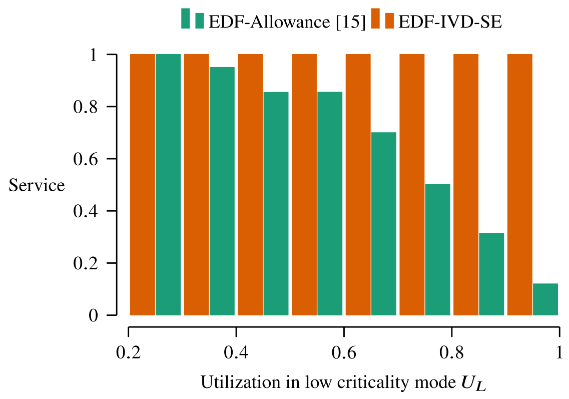


FIGURE 17. Time without deadline miss normalized to mission duration of EDF-IVD-SE and EDF-ALLOWANCE scheduled random task sets. While EDF-IVD-SE guarantees that the system survives the whole mission duration, EDF-Allowance can misses a deadline under bad circumstances. The likelihood that EDF-Allowance misses a deadline increases with utilization, and with the environment's error probability. Contrary, the service of EDF-IVD-SE is independent of the environment's error probability.

XI. CONCLUSION

Fault-tolerant real-time systems in emerging applications are not allowed to fail, but their budgets for error mitigation are minimal. We introduce and prove the feasibility of EDF-IVD-SE, our verification friendly mixed-criticality approach, which provides single error tolerance by design and guarantees that the most critical system functions are always operational.

By extensive simulation experiments we showed that EDF-IVD-SE has up to 1.56 better acceptance rate compared to similar state-of-the-art approaches, and how EDF-IVD-SE can provide full service to low criticality tasks for 1.93 times the original duration. Affordable and powerful, EDF-IVD-SE is a suitable mixed-criticality scheduling approach for fault-tolerant real-time systems in emerging critical applications.

APPENDIX

A. GLOSSARY

See Table 4.

B. EDF-NUVD PROOF

Proposition 11 (EDF-NUVD schedulability [17]): Let τ be a [...][dual-criticality] task [...][set] and let $0 < x_i < 1$, for each [...][high criticality task]. If

$$U_L^L + \sum_{i:\chi_i=H} u_i^L/x_i \le 1 \quad \sum_{i:\chi_i=H} u_i^H/(1-x_i) \le 1 \tag{31}$$

then τ is schedulable by EDF-NUVD.

TABLE 4. Notation.

	Name	Description
A	allowance	Tolerable time beyond overrun without compromising any deadlines
α	arrival time	Time where job enters the scheduler's queue
χ	criticality	Required level of assurance against failure
H	high criticality level	
L	low criticality level	
D	absolute deadline	Time for a job to finish
$\hat{D}$	virtual absolute deadline	Virtual earlier time for a job to finish
d	relative deadline	Duration for a job to finish
$\hat{d}$	virtual relative deadline	Virtual shorter duration for a job to finish
p	error probability	Probability for an error
γ	execution time	Time it takes to finish the computation without interruption
c	WCET parameter	Maximum execution time for a job which does not trigger a mode change
J	job	Computation
t^{zZz}	last idle point	latest time where a job from a high criticality task did not execute in low criticality mode
n_H	number of high criticality tasks	Number of tasks in task set with high criticality
t^*	first overrun time	First time where a job from a high criticality task exceeds its low criticality execution time budget
$t^{\circledast}$	second overrun time	Second time where a job from a high criticality task exceeds its low criticality execution time budget
p	period	Minimum interarrival time between two jobs of the same task
z	pessimism	Pessimism in estimating the task's WCET
x	virtual deadline scaling factor	Factor multiplied with relative deadline to get virtual relative deadline

Proof: No deadline is missed by EDF-NUVD in low criticality mode if Eq. (5) holds. The first time where a job from a high criticality task exceeds its low criticality WCET parameter is at t^*. Consider a job J_{ij} of a high criticality task

TABLE 4. ***(Continued.)*** **Notation.**

	Name	Description
σ	supply	Computation resources provisioned by CPU
τ	task	Reoccuring computation
u	task utilization	parameter over period
$\mathbb{T}$	task set	Independent reoccuring computations sharing the same processor
U	utilization	Sum of all task's WCET parameter over period
$\hat{U}$	virtual utilization	Sum of all task's WCET parameter over virtual relative deadline
$\tilde{U}$	increased virtual utilization	Sum of all task's WCET parameter over virtual relative deadline
W^J	job progression window	Window of possible progressions in executing a job

TABLE 5. **Abbreviations.**

CCDF	Complementary cumulative distribution function
COTS	Commercial off-the-shelf
CPU	Central processing unit
DBF	Demand bound function
EDF	Earliest deadline first
EDF-Allowance	Earliest deadline first with allowance
EDF-IVD	Earliest deadline first with improved virtual deadlines
EDF-IVD-SE	Earliest deadline first with improved virtual deadlines for single errors
EDF-NUVD	Earliest deadline first with non-uniform virtual deadlines
EDF-NUVD-SE	Earliest deadline first with non-uniform virtual deadlines for single errors
EDF-VD	Earliest deadline first with virtual deadlines
EDF-VD-SE	Earliest deadline first with virtual deadlines for single errors
NLP	Nonlinear programming
QoS	Quality of service
SLSQP	Sequential least squares programming
UML	Unified modeling language
WCET	Worst-case execution time

τ_i that is active at t^*. The job arrives at α_{ij}, and has a absolute deadline at $D_{ij} = \alpha_{ij} + d_i$. Before t^*, J_{ij} is EDF-scheduled according to its virtual absolute deadline $\hat{D}_{ij} = \alpha_{ij} + x_i d_i$.

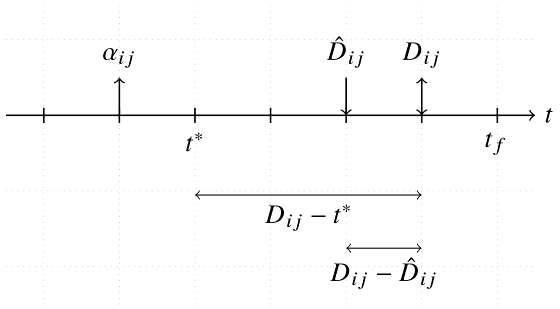


FIGURE 18. **Visualization of Proposition 11.**

In low criticality mode, all jobs would meet their virtual relative deadlines. Since J_{ij} is still active at t^*, its earliest virtual absolute deadline is at the first overrun time: $\hat{D}_{ij} \geq t^*$. Therefore the duration between the first overrun time and absolute deadline is larger or equal to the duration between absolute deadline and virtual absolute deadline, as shown in Fig. 18:

$$D_{ij} - t^* \geq D_{ij} - \hat{D}_{ij} \tag{32}$$

The worst case assumption is that each high criticality task has an active job with execution time equal to its WCET parameter in high criticality mode, and the execution needs to take place during $D_{ij} - \hat{D}_{ij}$. Therefore Eq. (6) regards the task set after t^* as a single-criticality task set under the worst case assumption, where each high-criticality task τ_i has period $D_{ij} - \hat{D}_{ij} = d_i(1 - x_i)$ and increased task utilization $u_i^H/(1 - x_i)$. If the sum over all increased task utilizations is below the supply of one, this single-criticality task set is schedulable. □

REFERENCES

[1] R. Wilhelm, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, P. Stenström, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, and R. Heckmann, "The worst-case execution-time problem—Overview of methods and survey of tools," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, pp. 1–53, Apr. 2008.

[2] S. Baruah and A. Burns, "Incorporating robustness and resilience into mixed-criticality scheduling theory," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2019, pp. 155–162.

[3] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 239–243.

[4] T. Fleming and A. Burns, "Incorporating the notion of importance into mixed criticality systems," in *Proc. 2nd Workshop Mixed Criticality Syst.*, 2014, pp. 33–38.

[5] D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi, "EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Nov. 2016, pp. 35–46.

[6] S. Baruah and P. Ekberg, "Graceful degradation in semi-clairvoyant scheduling," in *Proc. 33rd Euromicro Conf. Real-Time Syst.*, 2021, pp. 9:1–9:21.

[7] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 147–152.

[8] P. Huang, G. Giannopoulou, N. Stoimenov, and L. Thiele, "Service adaptions for mixed-criticality systems," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2014, pp. 125–130.

[9] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback–feedforward scheduling of control tasks," *Real-Time Syst.*, vol. 23, nos. 1–2, pp. 25–53, 2002.

[10] G. Chen, N. Guan, D. Liu, Q. He, K. Huang, T. Stefanov, and W. Yi, ''Utilization-based scheduling of flexible mixed-criticality real-time tasks,'' *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 543–558, Apr. 2018.

[11] K. Yang and Z. Guo, ''EDF-based mixed-criticality scheduling with graceful degradation by bounded lateness,'' in *Proc. IEEE 25th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2019, pp. 1–6.

[12] B. Ranjbar, B. Safaei, A. Ejlali, and A. Kumar, ''FANTOM: Fault tolerant task-drop aware scheduling for mixed-criticality systems,'' *IEEE Access*, vol. 8, pp. 187232–187248, 2020.

[13] B. Hu, L. Thiele, P. Huang, K. Huang, C. Griesbeck, and A. Knoll, ''FFOB: Efficient online mode-switch procrastination in mixed-criticality systems,'' *Real-Time Syst.*, vol. 55, pp. 471–513, Dec. 2018.

[14] B. Ranjbar, A. Hosseinghorban, M. Salehi, A. Ejlali, and A. Kumar, ''Toward the design of fault-tolerance-and peak-power-aware multi-core mixed-criticality systems,'' *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, May 21, 2021, doi: 10.1109/TCAD.2021.3082495.

[15] L. Bougueroua, L. George, and S. Midonnet, ''Dealing with execution-overruns to improve the temporal robustness of real-time systems scheduled FP and EDF,'' in *Proc. 2nd Int. Conf. Syst. (ICONS)*, Apr. 2007, p. 52.

[16] R. Schmidt and A. García-Ortiz, ''Service improvements in real-time uniprocessor scheduling with single errors,'' *IEEE Access*, vol. 9, pp. 43540–43550, 2021.

[17] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, ''Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems,'' *J. ACM*, vol. 62, no. 2, pp. 1–33, May 2015.

[18] C. L. Liu and J. W. Layland, ''Scheduling algorithms for multiprogramming in a hard-real-time environment,'' *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[19] L. George, P. Courbin, and Y. Sorel, ''Job vs. portioned partitioning for the earliest deadline first semi-partitioned scheduling,'' *J. Syst. Archit.*, vol. 57, no. 5, pp. 518–535, May 2011.

[20] D. Kraft, ''A software package for sequential quadratic programming,'' Deutsche Forschungs-und Versuchsanstalt für Luft-und Raumfahrt, Oberpfaffenhofen, Germany, Tech. Rep. DFVLR-FB 88-28, 1988.

[21] P. Virtanen *et al.*, ''SciPy 1.0: Fundamental algorithm for scientific computing in Python,'' *Nature Methods*, vol. 17, pp. 261–272, Feb. 2020.

[22] P. Huang, H. Yang, and L. Thiele, ''On the scheduling of fault-tolerant mixed-criticality systems,'' Comput. Eng. Netw. Lab., ETH Zürich, Zürich, Switzerland, Tech. Rep. 351, 2013.

[23] R. Schmidt and A. Garcia-Ortiz, ''Thready: A fast scheduling simulator for real-time task systems,'' in *Proc. 9th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Sep. 2020, pp. 1–6.

[24] E. Bini and G. C. Buttazzo, ''Measuring the performance of schedulability tests,'' *Real-Time Syst.*, vol. 30, nos. 1–2, pp. 129–154, May 2005.

[25] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, ''The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems,'' in *Proc. 24th Euromicro Conf. Real-Time Syst.*, Jul. 2012, pp. 145–154.

ROBERT SCHMIDT (Graduate Student Member, IEEE) received the Diploma degree in electrical engineering from the University of Bremen, Germany, in 2015. Form 2015 to 2017, he was worked as a Research Associate with the German Aerospace Center, Institute of Space Systems. Since 2017, he has been working as a Research Associate with the Institute of Electrodynamics and Microelectronics, University of Bremen.

ALBERTO GARCÍA-ORTIZ (Member, IEEE) received the Diploma degree in telecommunication systems from the Polytechnic University of Valencia, Spain, in 1998, and the Ph. D. degree *(summa cum laude)* from the Department of Electrical Engineering and Information Technology, Institute of Microelectronic Systems, Darmstadt University of Technology, Germany, in 2003. He worked at NewLogic, Austria, for two years. From 2003 to 2005, he worked as a Senior Hardware Design Engineer with the IBM Deutschland Research and Development, Böblingen. Then, he joined the start-up AnaFocus, Spain, where he was responsible for the design and integration of AnaFocus next generation Vision Systems-on-Chip. He is currently a Full Professor for the Chair of Integrated Digital Systems with the University of Bremen. His research interests include low-power design and estimation, communication-centric design, SoC integration, and variations-aware design. He received the Outstanding Dissertation Award from the European Design and Automation Association, in 2004, and the Innovation Award from IBM, in 2005, for his contributions to leakage estimation; he holds two issued patents for that work. He serves as an Editor for *JOLPE* and is a reviewer of several conferences, journals, and European projects.