Service Adaptions for Mixed-Criticality Systems

TIK Report No. 350

Pengcheng Huang, Georgia Giannopoulou,
Nikolay Stoimenov, Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zurich
firstname.lastname@tik.ee.ethz.ch

Abstract— Complex embedded systems are typically mixed-critical, where heterogeneous guarantees must be provided for functionalities of different criticalities. We study in this paper the reconfiguration of services provided to low criticality tasks in reaction to the overruns of high criticality tasks. We further investigate the quantification of the resetting time of the system services. For both service reconfiguration and resetting, we derive tight analysis results under Earliest Deadline First (EDF) scheduling.

1 Introduction

Complex embedded systems are typically mixed-critical – functionalities of different criticalities (importances) coexist on a common computing platform [17]. Examples can be seen in automotive applications (e.g. smartcar systems) and in avionics applications (e.g. flight control and management systems). For such systems, it is crucial to meet requirements of different degrees of rigorousness for different criticality levels.

A large body of research already exists on specifying and scheduling of Mixed-Criticality (MC) systems, see e.g. [19, 5, 13, 3, 8, 4]. The common practice is to model a single task on all different criticality levels, with possible different worst-case execution times depending on the criticality levels. Since the rigorousness and conservatism considered on a higher criticality is higher, the worst-case execution times are non-decreasing from high to low criticality levels. The online scheduler then decides which tasks to guarantee according to the actual runtimes of tasks: if any task exceeds its χ level worst-case execution time, then tasks of criticality levels no higher than χ are stopped after that. Various classical scheduling techniques are extended to support such dynamic behavior: Fixed Priority scheduling [19, 5], Earliest Deadline First (EDF) scheduling [13, 3, 8], Time-triggered scheduling [4].

On the one hand, it is often too pessimistic to reject all less critical tasks whenever a single high criticality task exceeds at runtime a certain execution

time threshold. Indeed, this has already been confirmed in [16, 18]. Here, instead of complete killing of less critical tasks, either partial killing or best-effort scheduling are adopted for the less critical tasks. However, the service that the system can still guarantee to the less critical tasks when a critical one overruns is not known. It remains *an open problem to compute bounds on the service that can be provided for the less critical tasks.* Such information could be used to guide the online reconfiguration of the provided system services.

On the other hand, the guarantees made by the above scheduling techniques are rarely acceptable in practice. Consider a typical avionic MC usecase – the Flight Management System (FMS) application. If a localization task (certified at level B, with A being the highest criticality level and E being the lowest according to the DO-178B standard [1]) exceeds a certain execution time threshold, we cannot simply reject the level C tasks (among them be the flightplan task) or perform best-effort scheduling for them, since the airplane constantly requires the flightplan information. Instead, a *degraded service* for the level C tasks should be guaranteed. Furthermore, for FMS, when the system can be *reset* to provide the original service to all tasks is an important measure to certify the runtime system. Therefore, offline analysis techniques need to be developed to bound the resetting time.

Contributions of this paper:

- We extend the EDF-VD [3] scheduling technique to guarantee a degraded service for the low criticality tasks when the high criticality tasks exceed their low criticality worst-case execution times.
- We extend the demand bound analysis for MC systems, which are scheduled by mode-switched EDF, see [8]. We present results on approximations of demand bounds for all tasks in different runtime modes.
- We propose an algorithm to compute the bounds on the degraded service provided to the low criticality tasks, as well as the configurations of EDF-VD to achieve such bounds.
- We present an analysis technique to bound offline the resetting time of the services provided to the low critical tasks. We demonstrate that a trade off between the degraded service and the resetting time exists.
- We show the applicability of our approach in a case study of an industrial application. For the considered Flight Management System application,the extended EDF-VD scheduling technique can schedule the application with a degraded service requirement for the low criticality tasks. Furthermore, service resetting time can be quantified, and traded for the degraded services of the LO criticality tasks.

Related Work The current research on mixed-criticality scheduling is initialized by the seminal work of Vestal et al. [19], where the common model for mixed-criticality systems is proposed. Various conventional scheduling

techniques are subsequently extended to the mixed-criticality domain, see [6] for a comprehensive survey. Motivated with a realistic avionics use-case, the work in this paper considers service degradation for low criticality tasks instead of rejecting them. In addition, the quantification of service resetting time is investigated. The closest work to ours is that presented in [8, 9], where a general mixed-criticality mode switch protocol is introduced and analyzed. Our work is different from [8, 9] in that an elaborated analysis is proposed to *compute* the degraded service for the low criticality tasks. To the best of our knowledge, the quantifications of the degraded service and the service resetting time have not been investigated in existing results on mixed-criticality scheduling.

The work in this paper is conducted in the mixed-criticality context, where real-time services for all tasks must be provided even under abrupt scenario changes. During such scenario changes, the service adaptions of different functionalities depend on their criticalities. Existing results on conventional adaptive real-time scheduling generally deviate from the assumptions made in a mixed-criticlaity setting: The works on mode-switched real-time scheduling either assume delayed system operation or task rejections, (see e.g. [14] for more references therein), and thus cannot directly apply in the mixed-criticality context. The classical overload scheduling techniques [15, 7] in general assume some online optimization to maximize the values of tasks which meet deadlines in reaction to system overloads. This is not suitable for mixed-criticality systems where deterministic real-time guarantees for all tasks need to be adjusted to the revealed scenarios. These works incorporating feedback control to realtime scheduling [12] generally can not make any deterministic guarantees in the system and is not suitable for mixed-criticality scheduling. Otherworks, e.g. adaptive fault-tolerant real-time scheduling [10, 11], reconfigure the system when faults occur in the system. However, this again deviates from the mixed-criticality assumption.

2 System Model

We consider a generalized mixed-criticality sporadic task model as proposed in [9]. Given is a dual-criticality task set $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ ($n \in \mathbb{N}$) scheduled on a uni-processor. Each task τ_i has a minimum inter-arrival time T_i and a relative deadline $D_i (D_i \leq T_i)$. The worst-case execution times (WCETs) of all tasks are modeled both on the HI (high) and on the LO (low) criticality levels. Only the HI criticality tasks are allowed to exceed their LO criticality WCETs at runtime. The system has *two operating modes*: whenever a HI criticality task exceeds its LO criticality WCET, the system transits *immediately* from the LO criticality mode to the HI criticality mode, during which all HI criticality tasks must finish their HI criticality

level WCETs before their deadlines, and the LO criticality tasks are possibly guaranteed with *degraded service parameters* (e.g. longer periods and/or deferred deadlines).

We can then abstract each task τ_i by a tuple, $\{\{T_i(\text{HI}), C_i(\text{HI}), D_i(\text{HI})\}, \{T_i(\text{LO}), C_i(\text{LO}), D_i(\text{LO})\}, \chi_i\}$:

- $T_i(\chi) \in \mathbb{R}^+$ is the minimum inter-arrival time of τ_i in mode χ.
- $C_i(\chi) \in \mathbb{R}^+$ is the WCET of τ_i in mode χ.
- $D_i(\chi) \in \mathbb{R}^+$ is the relative deadline of τ_i in mode χ.
- $\boldsymbol{\chi} = \{\text{HI}, \text{LO}\}$ is the set of criticality levels / operating modes.
- $\chi_i \in \boldsymbol{\chi}$ is the criticality level of task τ_i.

$\{T_i(\chi), C_i(\chi), D_i(\chi)\}$ characterizes the service to be guaranteed for τ_i in mode χ. A dual-criticality task set is said *schedulable* if there exists a scheduling technique, such that the service requirements for all tasks in both operating modes can be satisfied. For a HI criticality task τ_i, its WCET is adjusted from $C_i(\text{LO})$ to $C_i(\text{HI})$ when transiting to the HI criticality mode. This can lead to a unschedulable system: consider a scenario when a job of this task just finishes its low criticality WCET at its deadline, and at the same time a mode transition happens, this implies that this task must finish $C_i(\text{HI}) - C_i(\text{LO})$ units of execution in zero time. In order to schedule the HI criticality tasks, their deadlines need to be tuned in the LO criticality mode ($D_i(\text{LO}) \leq D_i(\text{HI})$), such that they can still meet their deadlines when transiting to the HI criticality mode. This concept has been explored in [3, 2, 8].

In summary, the following assumptions are made: if $\chi_i = \text{HI}$, then

$$T_i(\text{LO}) = T_i(\text{HI}) = T_i, C_i(\text{HI}) \geq C_i(\text{LO}), D_i(\text{LO}) \leq D_i(\text{HI}), \tag{1}$$

if $\chi_i = \text{LO}$, then

$$T_i(\text{LO}) = T_i, T_i(\text{LO}) \leq T_i(\text{HI}), C_i(\text{HI}) = C_i(\text{LO}), D_i(\text{LO}) \leq D_i(\text{HI}). \tag{2}$$

For the original EDF-VD scheduling technique [3, 2], all tasks are scheduled by Earliest Deadline First (EDF) in both HI and LO criticality modes (i.e. *mode-switched EDF*). All LO criticality tasks are immediately dropped when the system transits to the HI criticality mode. This can be viewed as a special case by setting $T_i(\text{HI}) := D_i(\text{HI}) := \infty$ for all LO criticality tasks. Furthermore, for notational convenience, we denote by τ_{HI} (τ_{LO}) the set of HI (LO) criticality tasks.

3 MC Service Reconfiguration

We present in this section the analysis of dual-criticality task sets scheduled by mode-switched EDF. We quantify the demand bounds of *all* tasks in the LO and HI criticality modes, and present the corresponding schedulability

test, both in Section 3.1. We present results on approximations of the analyzed demand bounds in Section 3.2. We view the service degradation of the LO criticality tasks by a scaling factor (≥ 1) of their periods and deadlines. For implicit-deadline task sets ($T_i = D_i$), an algorithm is proposed in Section 3.3 to adjust the services of the LO criticality tasks in the HI mode. The results presented extend existing works in [3, 8, 9].

3.1 MC Demand Bound Analysis

The demand bound of a task in a given interval is defined as the sum of execution times of all task instances, which have arrival times and deadlines both in this interval. During the LO criticality mode, the demand bound $\mathrm{dbf}_{\mathrm{LO}}$ of any task in an interval of length Δ can be bounded according to known results in [8]:

$$\mathrm{dbf}_{\mathrm{LO}}(\tau_i, \Delta) = \max\{\left\lfloor \frac{\Delta - D_i(\mathrm{LO})}{T_i(\mathrm{LO})} \right\rfloor + 1, 0\} \cdot C_i(\mathrm{LO}). \tag{3}$$

For the HI criticality mode, we need to consider the impacts of unfinished jobs at the transition point. Those are the jobs for which the schedules may be changed (i.e. tasks are scheduled by EDF and their deadlines may be adjusted at the time of mode switch).

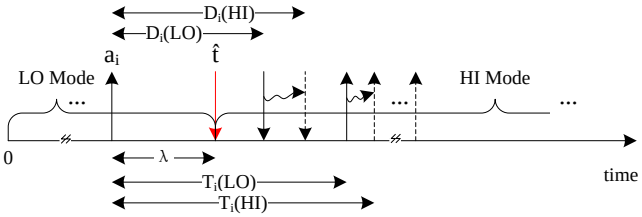

Figure 1: A mode transition triggered by task overrun

We depict in Fig. 1 a system undergoing a mode transition at time $\hat{t}$. A job of task τ_i arrives in the LO mode at time a_i. A mode transition is signaled λ units after that. From this time on, all jobs of this task are guaranteed with the HI criticality parameters.

In [8], only the demand bounds of the HI criticality tasks in the HI criticality mode are derived. This is done by identifying a worst-case demand bound including that of the unfinished job at the transition time (4). We proceed to present a general approach to analyze the demand bounds of *all* tasks in the HI criticality mode. Let us further define a set of functions (5)-(8):

$$\mathrm{RM}(\tau_i, \lambda) = C_i(\mathrm{HI}) - C_i(\mathrm{LO}) + \min\{D_i(\mathrm{LO}) - \lambda, C_i(\mathrm{LO})\}, \tag{4}$$

$$\mathrm{dbf}^1_{\mathrm{HI}}(\tau_i, \Delta) = \max\{\left\lfloor \frac{\Delta - D_i(\mathrm{HI})}{T_i(\mathrm{HI})} \right\rfloor + 1, 0\} \cdot C_i(\mathrm{HI}), \tag{5}$$

$$\mathrm{dbf}_{\mathrm{RM}}(\tau_i, \lambda, \Delta) = \begin{cases} \mathrm{RM}(\tau_i, \lambda) & \text{if } \Delta \geq D_i(\mathrm{HI}) - \lambda, \quad (6a) \\ 0 & \text{if } \Delta < D_i(\mathrm{HI}) - \lambda. \quad (6b) \end{cases}$$

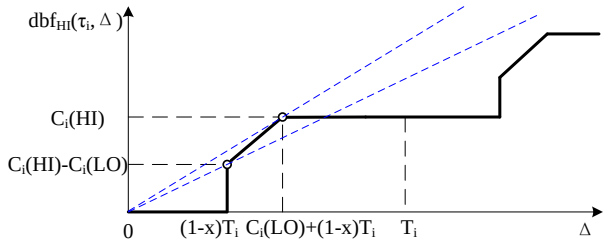

Figure 2: Approximation of $\mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta), \chi_i = \mathrm{HI}$

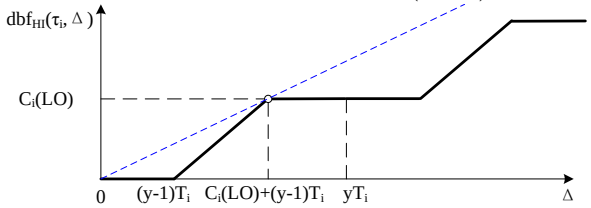

Figure 3: Approximation of $\mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta), \chi_i = \mathrm{LO}$

$$\mathrm{dbf}^2_{\mathrm{HI}}(\tau_i, \lambda, \Delta) = \mathrm{dbf}_{\mathrm{RM}}(\tau_i, \lambda, \Delta) + \max\{\left\lfloor \frac{\Delta - D_i(\mathrm{HI}) - (T_i(\mathrm{HI}) - \lambda)}{T_i(\mathrm{HI})} \right\rfloor + 1, 0\} \cdot C_i(\mathrm{HI}). \tag{7}$$

$$\mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta) = \sup\{\mathrm{dbf}^1_{\mathrm{HI}}(\tau_i, \Delta), \sup_{0 \le \lambda \le D_i(\mathrm{LO})} \{\mathrm{dbf}^2_{\mathrm{HI}}(\tau_i, \lambda, \Delta)\}\}. \tag{8}$$

Formally, we have the following result.

Lemma 3.1. The demand bound function of any task in the HI criticality mode can be calculated by (8).

Proof. All proofs are presented in the Appendix. □

Based on the above computed demand bounds, schedulability of a task set can be tested using existing results.

Theorem 3.1. [8] A dual-criticality task set τ is schedulable on an unit-speed processor, if $\forall \Delta \geq 0$:

$$\max\{\sum_{\tau} \mathrm{dbf}_{\mathrm{LO}}(\tau_i, \Delta), \sum_{\tau} \mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta)\} \leq \Delta. \tag{9}$$

3.2 MC Demand Bound Approximation

For the original EDF-VD scheduling technique, all LO criticality tasks are rejected in the HI criticality mode. The problem we are addressing in this paper is different, as we aim at computing the bounds on the service for the

LO criticality tasks in the HI criticality mode. This is not a trivial problem as both HI and LO criticality tasks change their schedules immediately when transiting to the HI criticality mode, which makes the analysis difficult. In order to simplify the problem, we restrict ourselves to consider implicit-deadline task sets ($D_i = T_i$). According to the discussions in Section 2, to ensure the schedulability of the HI criticality tasks in the HI criticality mode, their deadlines need to be tuned in the LO criticality mode. Similar to the original EDF-VD scheduling technique, we assume that the deadline tuning for the HI criticality tasks is characterized by a scaling factor x ($x \leq 1$):

$$T_i(\text{LO}) = T_i(\text{HI}) = D_i(\text{HI}), D_i(\text{LO}) = xD_i(\text{HI}). \tag{10}$$

In addition, for our problem, we assume the service degradation of the LO criticality tasks is characterized by another scaling factor y ($y \geq 1$):

$$T_i(\text{LO}) = D_i(\text{LO}), T_i(\text{HI}) = D_i(\text{HI}) = yT_i(\text{LO}). \tag{11}$$

We proceed to show that the demand bounds of all tasks in the HI criticality mode (Lemma 3.1) can be tightly approximated based on (8),(9). The essential idea is to bound for both HI criticality and LO criticality tasks, the nonlinear function $\text{dbf}_{\text{HI}}(\tau_i, \Delta)$ by certain slopes. The following results are presented.

Lemma 3.2. For a HI criticality task τ_i,

$$\text{dbf}_{\text{HI}}(\tau_i, \Delta) \leq \max\{\frac{C_i(\text{HI}) - C_i(\text{LO})}{(1-x)T_i}, \frac{C_i(\text{HI})}{C_i(\text{LO}) + (1-x)T_i}\} \cdot \Delta. \tag{12}$$

Lemma 3.3. For a LO criticality task τ_i,

$$\text{dbf}_{\text{HI}}(\tau_i, \Delta) \leq \frac{C_i(\text{LO})}{C_i(\text{LO}) + (y-1)T_i} \cdot \Delta. \tag{13}$$

We plot now in Fig. 2 and Fig. 3 the approximations of demand bounds in the HI criticality mode for both HI and LO criticality tasks (dashed lines).

3.3 MC Service Reconfiguration

We now show how to reconfigure the system such that a maximum degraded service for the LO criticality tasks can be guaranteed. Our analysis is based on the demand bound approximations shown in the previous section. For notational convenience, we define two functions as follows:

$$\begin{aligned} h(x) &= \sum_{\tau_{\text{HI}}} \frac{U_i(\text{HI})}{U_i(\text{LO}) + (1-x)}, \\ l(y) &= \sum_{\tau_{\text{LO}}} \frac{U_i(\text{LO})}{U_i(\text{LO}) + (y-1)}, \end{aligned} \tag{14}$$

where $U_i(\chi) = C_i(\chi)/T_i$. $h(x)$ ($l(y)$) represents the summed slopes of the HI (LO) criticality tasks in the HI criticality mode. We further use $U_{\chi 1}^{\chi 2}$

to denote $\sum_{\tau_i:\chi_i=\chi_1} U_i(\chi_2)$. We now explain the service reconfiguration in Algorithm 1. For the case when $U_{\mathrm{HI}}^{\mathrm{HI}} + U_{\mathrm{LO}}^{\mathrm{LO}} \leq 1$, the LO criticality tasks need not be reconfigured as the system can guarantee all tasks with their worst-case service requirements. For the case when $U_{\mathrm{HI}}^{\mathrm{LO}} + U_{\mathrm{LO}}^{\mathrm{LO}} > 1$, the system is even not schedulable during the LO criticality mode. Excluding the above conditions, the system needs to be further tested to see whether the reconfiguration of services in the HI criticality mode is feasible. This is ensured by enforcing that the total slopes of tasks in the HI criticality mode is ≤ 1 (line 8).

Algorithm 1: Service reconfiguration

```
   Input: τ
 1 if U_HI^HI + U_LO^LO ≤ 1 then
 2 |   x ← 1;
 3 |   y ← 1;
 4 else
 5 |   if U_HI^LO + U_LO^LO ≤ 1 then
 6 |   |   x ← U_HI^LO / (1 − U_LO^LO);
 7 |   |   if h(x) ≤ 1 then
 8 |   |   |   y ← inf{y ≥ 1 : h(x) + l(y) ≤ 1};
 9 |   |   else
10 |   |   |   return false;
11 |   |   end
12 |   else
13 |   |   return false;
14 |   end
15 end
16 return true;
```

Formally, we have the following result.

Theorem 3.2. Given a dual-criticality task set, Algorithm 1 can compute a minimized service degradation factor y for the LO criticality tasks, and a corresponding deadline tuning factor x for the HI criticality tasks, such that the task set is schedulable in all operating modes.

Example 3.1. Consider the set of dual-criticality task set as shown in Table 1.

According to Algorithm 1, we can derive $x = 0.5$, $y = 2.6488$. In practice, one can ceil the value of y and multiply the periods of the LO criticality tasks by 3 in the HI criticality mode. This can be done by the scheduler skipping 2 task instances in every 3 arrivals of a task. Furthermore, according to Theorem 3.2, there is a trade off between x and y. The more

Table 1: Example task set

τ	τ_1	τ_2	τ_3	τ_4	τ_5
χ	HI	LO	LO	LO	LO
T/D	60	8	30	90	15
C(HI)	18	4	4	6	3
C(LO)	3	4	4	6	3

we shorten the deadlines of the HI criticality tasks (providing that all tasks are still schedulable in the LO criticality mode), the better degraded service we can get for the LO criticality tasks. We plot in Fig. 4 the *ceiling* of y as a function of x. Notice that for $x < 0.5$ the system will not be schedulable in the LO criticality mode and for $x > 0.75$ the system will not be schedulable in the HI criticality mode even assuming all LO criticality tasks are rejected.

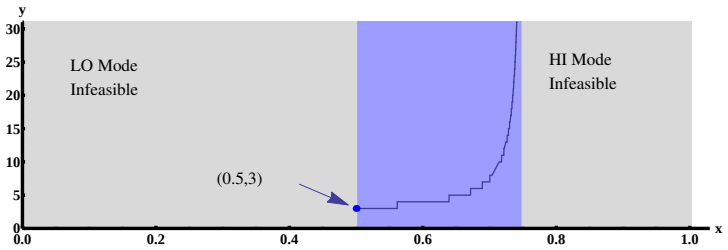

Figure 4: Trade off between x and y

4 MC Service Resetting

We present in this section how to quantify the resetting time of the system services (i.e. the elapsed time between when the system transits from the LO criticality mode to the HI criticality mode and when it can safely transit back). This is not a trivial problem since we do not statically know how many HI criticality tasks and for how long they will overrun. Our analysis in this section will not assume any such information. First, a sufficient condition for resetting the system service is given as follows.

Lemma 4.1. [16] A dual-criticality system can be safely reset to the LO criticality mode if the processor is idle.

Based on Lemma 4.1, a simple runtime mechanism can be implemented to reset the system to the LO criticality mode. However, it would be particularly interesting to quantify statically the worst-case resetting time, as

an important measure of the guarantees that a mixed-criticality scheduling algorithm can provide.

We use the same notion as shown in Fig. 1 (a system transits to the HI criticality mode at time $\hat{t}$). Furthermore, we define the arrival demand function of a task (adf) in the interval $[\hat{t}, \hat{t}+\Delta]$ as the cumulative execution times of all task instances *issued* within this interval. We define a list of functions as follows:

$$\begin{aligned}\mathrm{adf}^1_{\mathrm{HI}}(\tau_i, \lambda, \Delta) &= \mathrm{RM}(\tau_i, \lambda) \\ &+ \max\{\left\lceil \frac{\Delta - (T_i(\mathrm{HI}) - \lambda)}{T_i(\mathrm{HI})} \right\rceil, 0\} \cdot C_i(\mathrm{HI}),\end{aligned} \tag{15}$$

$$\mathrm{adf}^2_{\mathrm{HI}}(\tau_i, \Delta) = \left\lceil \frac{\Delta}{T_i(\mathrm{HI})} \right\rceil \cdot C_i(\mathrm{HI}), \tag{16}$$

$$\mathrm{adf}_{\mathrm{HI}}(\tau_i, \Delta) = \sup\{\mathrm{adf}^2_{\mathrm{HI}}(\tau_i, \Delta), \sup_{0 \le \lambda \le D_i(\mathrm{LO})} \{\mathrm{adf}^1_{\mathrm{HI}}(\tau_i, \lambda, \Delta)\}\}. \tag{17}$$

Formally, we have the following results.

Lemma 4.2. The arrival demand function of any task in the interval $[\hat{t}, \hat{t}+\Delta]$ can be calculated by (17).

Theorem 4.1. The service of the system can be reset at time $\hat{t}+\Delta_R$, where Δ_{R} is lower bounded by $\frac{\sum_{\tau} C_i(\chi_i)}{1-h(x)-l(y)}$.

Notice that according to Theorem 4.1, if we decrease x, then the resetting time will be reduced. Hence, one can simply set x as $\frac{U^{\mathrm{LO}}_{\mathrm{HI}}}{1-U^{\mathrm{LO}}_{\mathrm{LO}}}$, which is the minimum to guarantee the schedulability of tasks in the LO criticality mode. For setting y, there is a trade off between the resetting time and the degraded service in the HI criticality mode: if we increase the degraded service for the LO criticality tasks (i.e. decrease y), then the resetting time will be increased.

Example 4.1. Consider the same task set as shown in Example 3.1. We can now plot the service resetting time as a function of y.

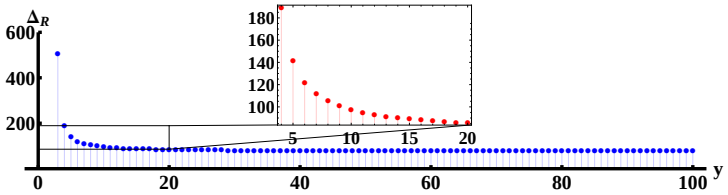

Figure 5: Trade off between Δ_R and y

As one can see, the resetting time decreases with increasing y. This gives us the flexibility to trade the degraded service of the LO criticality tasks for the resetting time. Furthermore, as y increases, the gain of saving in resetting time will also decrease.

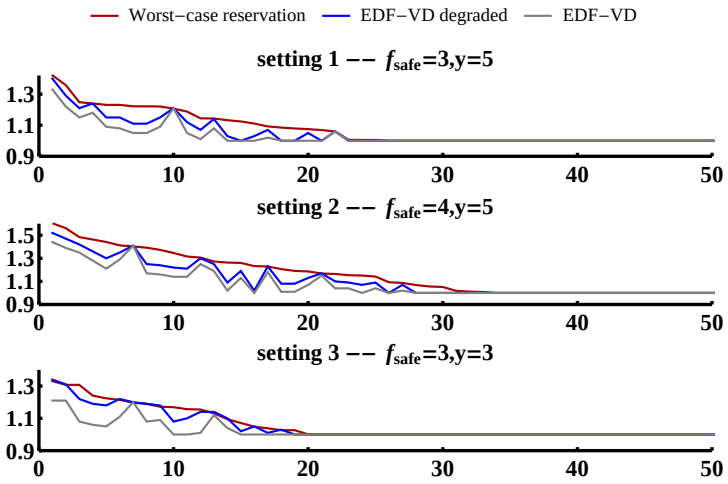

Figure 6: Processor speedup factors comparison, x-axis represents speedup factor, y-axis represents FMS instance

5 Case Study

Table 2: Task parameters for the FMS application

τ	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
T/D	5000	200	1000	1600	100	1000
C(LO)	$\{0, 20\}$	$\{0, 20\}$	$\{0, 20\}$	$\{0, 20\}$	$\{0, 20\}$	$\{0, 20\}$
χ	B	B	B	B	B	B
τ	τ_7	τ_8	τ_9	τ_{10}	τ_{11}	
T/D	1000	1000	1000	1000	1000	
C(LO)	$\{0, 20\}$	$\{0, 200\}$	$\{0, 200\}$	$\{0, 200\}$	$\{0, 200\}$	
χ	B	C	C	C	C	

We validate in this section our approach with an avionic usecase – the Flight Management System application (FMS). We show the applicability of our approach, such that service requirements for different runtime modes can be guaranteed. In addition, the service resetting time can be bounded in order to certify the scheduling of FMS.

We consider a subset of the original FMS, which contains the localization and the flightplan tasks (DO-178B level B and level C, where B corresponds to the HI criticality and C corresponds to the LO criticality). All tasks are abstracted as implicit-deadline sporadic tasks. The worst-case execution times (WCETs) of all tasks on level C are analyzed by an industrial partner and not available yet. However typical ranges can be assumed. We show the task parameters in Table 2 with timing information in units of ms. According to the discussions with our industrial partners, for avionics applications, a safety margin factor f_{safe} can be used to scale the WCETs on level C in order to get the WCETs on level B. We generate a set of FMS instances with random level C WCETs conforming to Table 2.

We compare three different approaches: EDF with worst-case reservation (i.e. all tasks are guaranteed with WCETs on the HI criticality level), EDF-VD with degraded service guarantee (i.e. a degraded service requirement for the LO criticality tasks is guaranteed in the HI criticality mode), and the original EDF-VD scheduling technique (all LO criticality tasks are rejected in the HI criticality mode). In all three cases, we calculate the minimum speedup factors of the processor (by linear search) such that guarantees as aimed by those approaches are provided for FMS.

We show in Fig. 6 the speedup factors of all 3 approaches for 50 randomly generated FMS instances. The results are shown for 3 settings as indicated in Fig. 6. Based on the results, the original EDF-VD scheduling technique always has the minimum speedup factors among all three approaches. This is intuitive since schedulability of the HI criticality tasks is guaranteed by complete rejection of the LO criticality tasks. However, EDF-VD is not applicable to FMS, as a degraded service requirement for the LO criticality tasks must be provided. One solution to achieve this is by EDF scheduling with worst-case reservation. However, as one can expect, this approach always has the maximum processor speedup factors among all three approaches. The extension of EDF-VD with degraded service guarantee has a processor speedup factor in between of the other two approaches. Based on those results, we conclude that resource efficiency can be achieved by our proposed method in comparison to worst-case reservation, while degraded service for the LO criticality tasks can be guaranteed in comparison to the original EDF-VD.

Furthermore, as shown in Fig. 6, if we increase f_{safe} (setting 2), then the processor speedup factor will also be increased (the maximum speedup factors of all three approaches in this case come close to 1.5, while the maximum speedup factors come close to 1.35 in setting 1). In addition, if we increase the degraded service for the LO criticality tasks (i.e. decrease y in setting 3), the extension of EDF-VD with degraded service guarantee always has close/equal processor speedup factors to the worst-case reservation approach. This implies that the gain in resource saving for our extension of EDF-VD decreases with decreasing y.

We continue to quantify the service resetting time for the FMS usecase. For this purpose, we pick one randomly generated FMS instance. We evaluate the impact of f_{safe} and y on the resetting time. The results are shown in Table 3 with resetting times given in units of second. For $f_{\text{safe}} = 3$, according to Algorithm 1, both x and y equal to 1. The FMS needs not be reconfigured in this case as the HI criticality WCETs of all tasks can be guaranteed. For $f_{\text{safe}} = 4$, the minimal y we calculate is 3, with a corresponding service resetting time of $21.6s$. If we increase y (i.e. decrease the degraded service for the LO criticality tasks), we can reduce the resetting time ($y = 4, \Delta_{\text{R}} = 7.8s$). If we continue to increase f_{safe} to 5, the minimum y becomes 22 in this case, which has an associated service resetting time of

Table 3: Resetting time

$f_{\text{safe}} = 3$			$f_{\text{safe}} = 4$			$f_{\text{safe}} = 5$		
y	x	Δ_{R}	y	x	Δ_{R}	y	x	Δ_{R}
1	1	0	3	0.25	21.6	22	0.25	2.1×10^3
-	-	-	4	0.25	7.8	23	0.25	661.8
-	-	-	5	0.25	5.92	24	0.25	406.1

$2.1 \times 10^3 s$.

6 Conclusion

We present in this paper the reconfiguration of the services provided to the low criticality tasks when the high criticality tasks overrun. Our scheduling algorithm is an extension of the well known EDF-VD scheduling technique. We extend the demand bound analysis for testing the schedulability of the system under this setting.Furthermore, we provide an analytical method to bound offline the resetting time of the services provided to the low criticality tasks. The presented techniques are validated with a realistic avionic application.

References

[1] RTCA/DO-178B, Software Considerations in Airborne Systems and Equipment Certification, 1992.

[2] 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 *ECRTS*, pages 145–154, 2012.

[3] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In *Algorithms - ESA*. 2011.

[4] S. Baruah and G. Fohler. Certification-cognizant time-triggered scheduling of mixed-criticality systems. In *RTSS*, pages 3–12, 2011.

[5] S. K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *RTSS*, pages 34–43, 2011.

[6] A. Burns and R. Davis. Mixed criticality systems-a review. 2013.

[7] G. Buttazzo, M. Spuri, and F. Sensini. Value vs. deadline scheduling in overload conditions. In *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pages 90–99. IEEE, 1995.

[8] P. Ekberg and W. Yi. Bounding and shaping the demand of mixed-criticality sporadic tasks. In *ECRTS*, pages 135–144, 2012.

[9] P. Ekberg and W. Yi. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. *Real-Time Systems*, pages 1–39, 2013.

[10] O. González, H. Shrikumar, J. A. Stankovic, and K. Ramamritham. Adaptive fault tolerance and graceful degradation under dynamic hard real-time scheduling. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 79–89. IEEE, 1997.

[11] J. Huang, J. Blech, A. Raabe, C. Buckl, and A. Knoll. Reliability-aware design optimization for multiprocessor embedded systems. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 239–246, 2011.

[12] C. Lu, X. Wang, and C. Gill. Feedback control real-time scheduling in orb middleware. In *Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings. The 9th IEEE*, pages 37–48. IEEE, 2003.

[13] T. Park and S. Kim. Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems. In *EMSOFT*, pages 253–262, 2011.

[14] J. Real and A. Crespo. Mode change protocols for real-time systems: A survey and a new proposal. *Real-time systems*, 26(2):161–197, 2004.

[15] P. Richardson and S. Sarkar. Adaptive scheduling: Overload scheduling for mission critical systems. In *Real-Time Technology and Applications Symposium, 1999. Proceedings of the Fifth IEEE*, pages 14–23. IEEE, 1999.

[16] F. Santy, L. George, P. Thierry, and J. Goossens. Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp. In *ECRTS*, pages 155–165, 2012.

[17] L. Sha. Resilient mixed-criticality systems. *The Journal of Defence Software Engineering*, 2009.

[18] H. Su and D. Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *DATE*, pages 147–152, 2013.

[19] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *RTSS*, pages 239–243, 2007.

7 Appendix

Proof. Lemma 3.1

Let us consider two different cases.

1- $\lambda \in [0, D_i(\text{LO})]$: In this case, the current job of τ_i may have not finished its LO criticality WCET, the worst-case left-overs of the current job can be bounded by $\min\{D_i(\text{LO}) - \lambda, C_i(\text{LO})\}$. Since starting from $\hat{t}$ the current and

future jobs of τ_i execute according to the HI criticality mode parameters, the total left-overs of the current job of τ_i can be bounded by (4). We have to further identify a time interval of length Δ within $[\hat{t}, +\infty)$, which has the worst-case demand bound. Suppose that such an interval is represented as $[\hat{t}+\eta, \hat{t}+\eta+\Delta]$ ($\eta \geq 0$), i.e. the interval starts η units of time after $\hat{t}$. Let us consider further two different cases:

• $\eta > 0$: In this case, the left-over job does *not* belong to the interval and its demand is not considered. Let γ represent the value of $((\lambda+\eta) \bmod T_i(\mathrm{HI}))$, then the number of maximum complete arrivals of τ_i in $[\hat{t}+\eta, \hat{t}+\eta+\Delta]$ is bounded by:

$$\begin{cases} \max\{\left\lfloor \dfrac{\Delta - D_i(\mathrm{HI}) - (T_i(\mathrm{HI}) - \gamma)}{T_i(\mathrm{HI})} \right\rfloor + 1, 0\} \text{ if } 0 < \gamma, & \text{(18a)} \\ \max\{\left\lfloor \dfrac{\Delta - D_i(\mathrm{HI})}{T_i(\mathrm{HI})} \right\rfloor + 1, 0\} \qquad \text{if } \gamma = 0. & \text{(18b)} \end{cases}$$

Hence, when $\gamma = 0$ we can get the maximum complete arrivals of τ_i within such an interval. And the demand bound of τ_i in $[\hat{t}+\eta, \hat{t}+\eta+\Delta]$ is upper bounded by (5).

• $\eta = 0$: In this case the left-over job needs to be considered for the demands in interval $[\hat{t}, \hat{t}+\Delta]$. Notice further that the left-over job is only considered when $\Delta \geq D_i(\mathrm{HI}) - \lambda$, hence the demand of this job is given by (4).

The maximum number of future complete arrivals of τ_i within$[\hat{t}, \hat{t}+\Delta]$ is bounded by:

$$\max\{\left\lfloor \frac{\Delta - D_i(\mathrm{HI}) - (T_i(\mathrm{HI}) - \lambda)}{T_i(\mathrm{HI})} \right\rfloor + 1, 0\}. \tag{19}$$

Hence the demand bound function of τ_i can be given by (7).

2- $\lambda \in (D_i(\mathrm{LO}), T_i(\mathrm{LO})]$: In this case τ_i has no active instance running in the system, we can derive similarly that the maximum number of complete arrivals of τ_i within an interval of length Δ happens when this interval starts with an arrival of τ_i. This number is again bounded by (18b). Hence the demand bound function in this case can be represented by (5).

Now, the demand bound function for any task τ_i in the HI criticality mode can be represented by (8). □

Proof. Lemma 3.2

Let us consider two different cases.

1- $0 \leq \Delta \leq T_i$: In this case, according to (5), (7), (8), we can get:

- $0 \leq \Delta < (1-x)T_i$: In this case, both $\mathrm{dbf}^1_{\mathrm{HI}}(\tau_i, \Delta)$ and $\mathrm{dbf}^2_{\mathrm{HI}}(\tau_i, \lambda, \Delta)$ (independent of λ) evaluate to zero. Hence, the demand bound is constantly zero.
- $(1-x)T_i \leq \Delta \leq C_i(\mathrm{LO}) + (1-x)T_i$: In this case, $\mathrm{dbf}^1_{\mathrm{HI}}(\tau_i, \Delta)$ evaluates to zero. $\mathrm{dbf}^2_{\mathrm{HI}}(\tau_i, \lambda, \Delta)$ has the maximum value $C_i(\mathrm{HI}) - C_i(\mathrm{LO}) + \Delta - (1-x)T_i$ when $\lambda = T_i - \Delta$. Hence, the worst-case demand bound is $C_i(\mathrm{HI}) - C_i(\mathrm{LO}) + \Delta - (1-x)T_i$.

- $C_i(\text{LO})+(1-x)T_i \leq \Delta \leq T_i$: In this case, $\text{dbf}^1_{\text{HI}}(\tau_i, \Delta)$ evaluates to $C_i(\text{HI})$. The worst-case of $\text{dbf}^2_{\text{HI}}(\tau_i, \lambda, \Delta)$ also evaluates to $C_i(\text{HI})$ (when $\lambda \leq xT_i - C_i(\text{LO})$). Hence, the maximum demand bound is $C_i(\text{HI})$.

2- $(k-1)T_i \leq \Delta \leq kT_i$ $(k \in \mathbb{N}^+)$: Assume that $\Delta = (k-1)T_i + \delta$ $(0 \leq \delta \leq T_i)$. First, we can derive that:

$$\text{dbf}^1_{\text{HI}}(\tau_i, \Delta) = (k-1)C_i(\text{HI}) + \text{dbf}^1_{\text{HI}}(\tau_i, \delta). \tag{20}$$

Second, $\text{dbf}^2_{\text{HI}}(\tau_i, \lambda, \Delta)\}$ achieves the maximum value when $\lambda = T_i - \delta$, then:

$$\begin{aligned} &\sup_{0\leq\lambda\leq xT_i} \{\text{dbf}^2_{\text{HI}}(\tau_i, \lambda, (k-1)T_i + \delta)\} \\ &= \text{dbf}^2_{\text{HI}}(\tau_i, T_i - \delta, (k-1)T_i + \delta) \\ &= (k-1)C_i + \text{dbf}^2_{\text{HI}}(\tau_i, T_i - \delta, \delta) \\ &= (k-1)C_i + \sup_{0\leq\lambda\leq xT_i} \{\text{dbf}^2_{\text{HI}}(\tau_i, \lambda, \delta)\}. \end{aligned} \tag{21}$$

Hence,

$$\begin{aligned} \text{dbf}_{\text{HI}}(\tau_i, \Delta) =& \sup\{\text{dbf}^1_{\text{HI}}(\tau_i, \Delta), \sup_{0\leq\lambda\leq xT_i} \{\text{dbf}^2_{\text{HI}}(\tau_i, \lambda, \Delta)\}\} \\ =& (k-1)C_i(\text{HI}) + \text{dbf}_{\text{HI}}(\tau_i, \delta). \end{aligned} \tag{22}$$

Using the above results, we can draw the HI mode demand bound curve as shown in Fig. 2, and approximate it by two linear functions (dotted lines in Fig. 2). □

Lemma 3.3. This can be similarly derived as shown in Lemma 3.2. □

Proof. Theorem 3.2

The schedulability of a task set in the LO criticality mode is tested by line 5 in Algorithm 1. We have to enforce further the schedulability in the HI mode according to Theorem 3.1. According to Lemma 3.2 and Lemma 3.3:

$$\begin{aligned} &\sum_{\tau} \text{dbf}_{\text{HI}}(\tau_i, \Delta) \leq \Delta \\ \Leftarrow &\sum_{\tau_{\text{HI}}} \max\{\frac{C_i(\text{HI}) - C_i(\text{LO})}{(1-x)T_i}, \frac{C_i(\text{HI})}{C_i(\text{LO}) + (1-x)T_i}\} \\ &+ \sum_{\tau_{\text{LO}}} \frac{C_i(\text{LO})}{C_i(\text{LO}) + (y-1)T_i} \leq 1 \\ \Leftrightarrow &h(x) + l(y) \leq 1. \end{aligned} \tag{23}$$

If all LO criticality tasks are rejected in the HI criticality mode, the inequality becomes $h(x) \leq 1$, which is a sufficient condition for all HI criticality tasks to be schedulable in the HI criticality mode by rejecting all LO criticality tasks. Furthermore, we observe that the minimal value of y increases with increasing x. Hence, we can set x to $\frac{U^{\text{LO}}_{\text{HI}}}{1-U^{\text{LO}}_{\text{LO}}}$ (the minimum x to guarantee schedulability in the LO criticality mode) in order to get the maximum degraded service for the LO criticality tasks (minimal y): $y = \inf\{y \geq 1 : h(x) + l(y) \leq 1\}$. □

Proof. Lemma 4.2

This lemma can be similarly derived as shown in the proof of Lemma 3.1.

1-$\lambda \in [0, D_i(\text{LO})]$: In this case, the remanning execution demand of the current job needs to be counted (Equation 4). The number of future arrivals within $[\hat{t}, \hat{t}+\Delta]$ can be bounded by: $\max\{\left\lceil \frac{\Delta-(T_i(\text{HI})-\lambda)}{T_i(\text{HI})} \right\rceil, 0\}$. Hence, the total arrived demands in $[t, t+\Delta]$ can be given by (15).

2-$\lambda \in (D_i(\text{LO}), T_i(\text{LO})]$: Clearly, in this case, there is no unfinished instance of τ_i in the system. We get the worst-case arrived demands within $[\hat{t}, \hat{t}+\Delta]$ when t coincides with an arrival of τ_i, which is bounded by (16).

Summarizing: the worst-case arrived demands within $[\hat{t}, \hat{t}+\Delta]$ is given by (17). □

Proof. Theorem 4.1

We identify a processor idle time after $\hat{t}$, $\hat{t}+\Delta_{\text{R}}$, at which time the system can be safely reset to the LO criticality mode. It then must be the case that $\sum_{\tau} \text{adf}_{\text{HI}}(\tau_i, \Delta_{\text{R}}) \leq \Delta_{\text{R}}$.

To show a lower bound of Δ_{R}, similar to Lemma 3.2 and Lemma 3.3, we can first approximate the arrival demand functions for both the HI and LO criticality tasks in the interval $[\hat{t}, \hat{t}+\Delta]$:

$$\begin{aligned} \text{adf}_{\text{HI}}(\tau_i, \Delta) &\leq C_i(\text{HI}) + \frac{C_i(\text{HI})}{C_i(\text{LO}) + (1-x) \times T_i} \times \Delta \quad \text{if } \chi_i = \text{HI}, \\ \text{adf}_{\text{HI}}(\tau_i, \Delta) &\leq C_i(\text{LO}) + \frac{C_i(\text{LO})}{C_i(\text{LO}) + (y-1) \times T_i} \times \Delta \quad \text{if } \chi_i = \text{LO}. \end{aligned} \tag{24}$$

It follows that:

$$\begin{aligned} & \sum_{\tau} \text{adf}_{\text{HI}}(\tau_i, \Delta_{\text{R}}) \leq \Delta_{\text{R}} \\ \Leftarrow & \sum_{\tau} C_i(\chi_i) + \left(\sum_{\tau_{\text{HI}}} \frac{C_i(\text{HI})}{C_i(\text{LO}) + (1-x) \times T_i} \right. \\ & \left. + \sum_{\tau_{\text{LO}}} \frac{C_i(\text{LO})}{C_i(\text{LO}) + (y-1) \times T_i} \right) \times \Delta_{\text{R}} \leq \Delta_{\text{R}} \\ \Leftrightarrow & \Delta_{\text{R}} \geq \frac{\sum_{\tau} C_i(\chi_i)}{1 - \frac{U_{\text{HI}}^{\text{HI}}}{1-x} - \frac{U_{\text{LO}}^{\text{LO}}}{y-1}}. \end{aligned} \tag{25}$$

□