Mixed-Criticality Scheduling with Varying Processor Supply in Compositional Real-Time Systems

Kecheng Yang
Department of Computer Science
Texas State University
yangk@txstate.edu

Zheng Dong
Department of Computer Science
Wayne State University
dong@wayne.edu

***Abstract*—In order to mitigate the pessimism of parameter estimation in real-time systems, mixed-criticality (MC) scheduling has been proposed and studied. In light of the first MC scheduling work focusing on multiple estimations on the worst-case execution times (WCETs), a few following works also have extended this approach to other dimensions, such as period, relative deadlines, and processor speeds. Nonetheless, in most existing work on MC scheduling, a flat-structured scheduling approach, whereas compositional real-time systems with hierarchical scheduling are of great interest, especially for large-scale real-time systems. In this work, we aim to extend the fundamental ideas and frameworks of MC scheduling to another dimension, namely the budget estimation. To further illustrate this approach, we form up a specific scheduling problem in the context of a single virtual processor characterizing by the periodic resource model and propose a virtual-deadline-based algorithm to solve it.**

***Index Terms*—real-time systems, mixed-criticality scheduling, hierarchical scheduling, periodic resource model**

I. Introduction

Real-time systems design is often aiming for providing *guarantees* for meeting deadlines *in all possible scenarios*. As a result, significant *pessimism* usually exists in real-time scheduling analysis and design. The system parameters, *e.g.*, the needed execution time for a piece of code, are provisioned as upper bounds, which are often not tight, on the worst case. Consequently, while the system design and certification need to follow these pessimistic provisioned system parameters, computing resources might be significantly underutilized in practice due to the potentially huge gap between the general scenario during runtime and the worst-case provision used for analysis, design, and certification. One approach to mitigate such pessimism is *mixed-criticality (MC) scheduling* [40]. Under MC scheduling, tasks are grouped to two distinct *criticality level*—HI (stands for high criticality) and LO (stands for low criticality), and each system parameter might have two provisions—a more pessimistic one that absolutely upper bounds the worst case scenario, and a less pessimistic one that covers a dominant majority of scenarios in practice (*e.g.*, take the observed worst case in measurement-based experiments). An MC scheduler is designed based on the given criticality levels and system provisions, and needs to guarantee that the deadlines of all tasks (*i.e.*, both HI- and LO-tasks) are met in "normal" scenarios in common practice while the deadlines of all HI-tasks are still met even if any pathological extreme cases do happen.

Although MC scheduling has received much attention in the real-time systems research community, most existing work has been directed to a flat-structured scheduling approach, *i.e.*, a single central scheduler is used to schedule all tasks in the entire system. However, as computing systems, even embedded ones, become increasingly complicated and highly integrated, a single system may need to be designed, analyzed, implemented, and certified as several isolated *components*, with each component having the "illusion" of executing on a dedicated *virtual* platform [14]. In a compositional real-time system, the scheduling follows a hierarchical approach. An upper-level scheduler distributes the computing capacity of the physical computing platform to each component and determines the characteristics of the virtual platform in each component. Then, each component has a lower-level scheduler to schedule the tasks in that component upon that virtual platform.

In order to analyze and certify each component independently, interfaces are needed to characterize the *supply* provided by a virtual processor (VP), *i.e.*, available processing time units from the physical processor to support task execution. For example, the *periodic resource model* is such a fundamental interface, which characterizes a VP by a pair of parameters (Π, Θ), with the interpretation that *at least* Θ time units of processor time is guaranteed to the supported task set every Π time units.

As Θ indicates the *minimum* budget for *any* resource period, it may be necessarily estimated with significant pessimism, just like what happens to the *worst-case execution time (WCET)* estimation. As a result, the resource may be greatly wasted in the actual runtime.

Contributions. In this paper, we extend the work of mixed-criticality scheduling to the resource supply estimation in the context of compositional real-time systems. By provisioning resource budget with multiple estimations, we show that fundamental ideas and framework of mixed-criticality scheduling can be applied in another dimension of scheduling problems that have not been considered before. To further illustrate this approach, we form up a specific scheduling problem in the context of a single virtual processor characterizing by the periodic resource model and propose a virtual-deadline-based algorithm to solve it.

Organization. In the rest of this paper, we describe our system model and review a few direct background results (Sec. II), describe our proposed scheduling algorithm (Sec. III), present a corresponding schedulability test (Sec. IV), discuss related work (Sec. V), and conclude (Sec. VI).

II. System Model and Background

In this paper, we consider the preemptive scheduling of a set of tasks of two criticalities on a single VP. Also, we assume time is discrete in this paper, *i.e.*, all parameters representing amount of time units are assumed to be integers and any scheduling event and decision must happen at an integer time instant.

Resource model. Similar to the *periodic resource model* [35], we assume this single virtual processor provides certain available processing time units to the task set every Π time units, and Π is called the *resource period* of this virtual processor. The amount of available time units within every resource period is called the *budget* of this virtual processor. In the original periodic resource model, a single budget parameter Θ is assumed, indicating the minimum amount of budget in any resource period. In contrast, we apply two estimations to the budget: a *critical* budget Θ^C indicating an absolute lower bound on budget in any resource period (*e.g.*, derived by the most pessimistic analysis), and a *nominal* budget Θ^N indicating a less pessimistic lower bound on budget in any resource period (*e.g.*, based on observations in empirical experiments). That is, $\Theta^C \leq \Theta^N$.

Task model. On this virtual processor, a set of sporadic tasks $\mathcal{T}$ is supposed to be scheduled. Each task $\tau_i \in \mathcal{T}$ releases a sequences of *jobs* with a minimum separation of T_i time units, where T_i is called the *period* of τ_i. Any job of τ_i may be executed for up to C_i time units to complete, *i.e.*, C_i is the WCET of τ_i. In this paper, we also assume that the deadlines are implicit, *i.e.*, every task τ_i has a *relative deadline* of T_i time units indicating that every job of τ_i has an *absolute deadline* at T_i time units after its release. Furthermore, the *utilization* of task τ_i is defined by $u_i = C_i/T_i$. Each task must be specified as either a *high-critical (*HI*)* or *low-critical (*LO*)* task. We also denote the set of HI-tasks and LO-tasks by $\mathcal{T}_{\text{HI}}$ and $\mathcal{T}_{\text{LO}}$, respectively. That is, $\mathcal{T}_{\text{HI}} \cup \mathcal{T}_{\text{LO}} = \mathcal{T}$ and $\mathcal{T}_{\text{HI}} \cap \mathcal{T}_{\text{LO}} = \emptyset$. We also denote the total utilization of all tasks, HI-task, and LO-tasks, respectively, as follows:

$$U = \sum_{\tau_i \in \mathcal{T}} u_i, \quad U_{\text{HI}} = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i, \quad U_{\text{LO}} = \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i.$$

We also call a job of a HI-task a HI-job for short and call a job of a LO-task a LO-job for short, respectively. Also, a job is *pending* if it is released and has not completed.

Schedulability criteria. Note that, in this paper, two estimations on the budget (Θ^C and Θ^N) for every resource period Π have been applied, while the actual budget during runtime is unknown for the pre-runtime analysis. Therefore, the *schedulability* of the system is defined as

- the deadlines of all (HI- and LO-) tasks must be met, if at least Θ^N available time units are provided as the budget during *every* resource period;
- the deadlines of all HI-tasks must be met, if less than Θ^N (but still at least Θ^C) available time units are provided during *some* resource period.

In the rest of this section, we review algorithm EDF-VD and the periodic resource model in more detail.

A. Algorithm EDF-VD

The scheduling algorithm EDF-VD was first proposed for scheduling implicit-deadline mixed-criticality tasks with multiple *execution time* estimations on a *dedicated* uniprocessor [5, 6].

Under EDF-VD, each HI-job is assigned a virtual deadline earlier than its actual deadline. Specifically, a scaling factor $0 < x \leq 1.0$ is applied system-widely to all HI-tasks so that each HI-task τ_i has a relative deadline $T'_i = x \cdot T_i$. On the other hand, every LO-job is assigned a virtual deadline equal to its actual deadline.

During runtime, EDF-VD starts with scheduling all jobs with respect to their *virtual* deadlines — the earlier the virtual deadline, the higher the priority. If all HI-jobs complete their execution by their less pessimistic execution time estimation, all virtual deadlines and therefore all actual deadlines are guaranteed to be met. If any HI-job executes over its less pessimistic execution time estimation without signaling completion, all LO-jobs are dropped by EDF-VD immediately and afterwards, all HI-jobs are scheduled by EDF with respect to their *actual* deadlines.

The setting of the scaling factor x and utilization-based schedulability tests have been investigated in [5, 6], but we omit the details here due to the difference of system model.

B. Periodic Resource Model

In the periodic resource model [35], a VP is characterized by two parameters (Π, Θ), which indicate that this VP supplies Θ units of processor time every Π time units, where $0 < \Theta \leq \Pi$.

Note that, a VP corresponding to a dedicated physical processor that is always available is a special case in the periodic resource model where $\Theta = \Pi$.

The *supply bound function (SBF)* of the VP, denoted $\mathsf{sbf}(t)$, indicates the *minimum* processor time this VP can supply during any time interval of length t. Shin and Lee [35] have shown that $\mathsf{sbf}(t)$ can be calculated by

$$\mathsf{sbf}(t) = \begin{cases} 0 & \text{if } t' < 0 \\ \left\lfloor \frac{t'}{\Pi} \right\rfloor \cdot \Theta + \epsilon & \text{if } t' \geq 0 \end{cases}$$

where

$$t' = t - (\Pi - \Theta),$$

$$\epsilon = \max\left(t' - \Pi \left\lfloor \frac{t'}{\Pi} \right\rfloor - (\Pi - \Theta), 0\right).$$

This definition reflects the worst-case scenario illustrated in Figure 1. In [35], a *linear* supply bound function, which is a

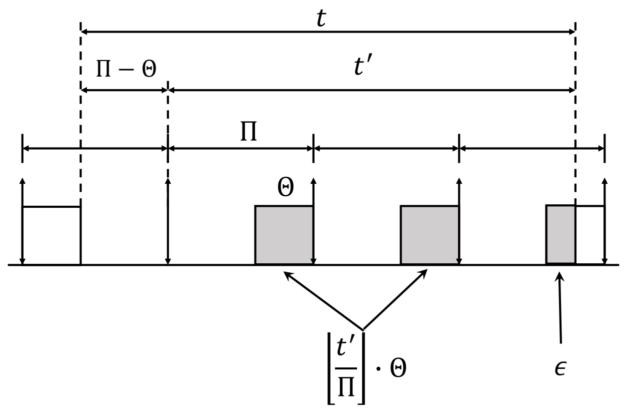

Fig. 1: Worst-case supply of a periodic resource.

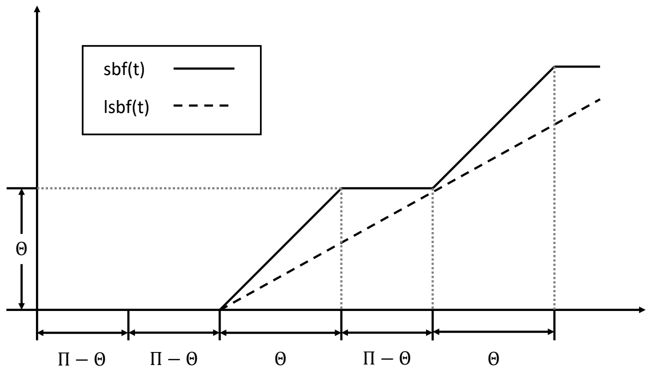

Fig. 2: An illustration for functions $\mathsf{sbf}(t)$ and $\mathsf{lsbf}(t)$.

lower-bound on the corresponding supply bound function, is also defined by

$$\mathsf{lsbf}(t) = \frac{\Theta}{\Pi}(t - 2(\Pi - \Theta)).$$

That is, $\forall t, \mathsf{lsbf}(t) \leq \mathsf{sbf}(t)$. Fig. 2 illustrates the functions $\mathsf{sbf}(t)$ and $\mathsf{lsbf}(t)$. On the other hand, under EDF scheduling, the *demand bound function* of a task τ_i is calculated by

$$\mathsf{dbf}(t, \tau_i) = \left\lfloor \frac{t}{T_i} \right\rfloor \cdot C_i,$$

and the demand bound function for a task set $\mathcal{T}$ is

$$\mathsf{dbf}(t, \mathcal{T}) = \sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}(t, \tau_i).$$

Linear demand bound functions, which are an upper-bound on the corresponding demand bound functions, are also defined by

$$\mathsf{ldbf}(t, \tau_i) = u_i \cdot t, \text{ and}$$

$$\mathsf{ldbf}(t, \mathcal{T}) = \left(\sum_{\tau_i \in \mathcal{T}} u_i \right) \cdot t.$$

That is, $\forall t$ and $\mathcal{T}$, $\mathsf{dbf}(t, \mathcal{T}) \leq \mathsf{ldbf}(t, \mathcal{T})$. Thus, $\mathsf{ldbf}(t, \mathcal{T}) \leq \mathsf{lsbf}(t)$ implies $\mathsf{dbf}(t, \mathcal{T}) \leq \mathsf{sbf}(t)$. The following lemma and theorem have been shown in [35], and Theorem 1 below in fact provides a utilization-based schedulability test.

Lemma 1. *(Lemma 5 in [35]) Task set $\mathcal{T}$ is schedulable by EDF on a VP (Π, Θ) if $\mathsf{ldbf}(T^{\min}, \mathcal{T}) \leq \mathsf{lsbf}(T^{\min})$ where $T^{\min} = \min_{\tau_i \in \mathcal{T}} T_i$.*

Theorem 1. *(Theorem 7 in [35]) Task set $\mathcal{T}$ is schedulable by EDF on a VP (Π, Θ) if*

$$U \leq \frac{\Theta}{\Pi} \left(1 - \frac{2(\Pi - \Theta)}{T^{\min}} \right),$$

where $U = \sum_{\tau_i \in \mathcal{T}} u_i$ and $T^{\min} = \min_{\tau_i \in \mathcal{T}} T_i$.

III. Algorithm EDF-VDVP

In this section, we present our scheduling algorithm EDF-VDVP (stands for "EDF with virtual deadlines and varying processor supply"). By Theorem 1, we first require the following to hold as the basis of our utilization-based schedulability test:

$$U \leq \beta^N \text{ and } U_{\text{HI}} \leq \beta^C \tag{1}$$

where

$$\beta^N = \frac{\Theta^N}{\Pi} \left(1 - \frac{2(\Pi - \Theta^N)}{T^{\min}} \right) > 0, \tag{2}$$

$$\beta^C = \frac{\Theta^C}{\Pi} \left(1 - \frac{2(\Pi - \Theta^C)}{T^{\min}_{\text{HI}}} \right) > 0, \tag{3}$$

$$T^{\min} = \min_{\tau_i \in \mathcal{T}} T_i,$$

$$T^{\min}_{\text{HI}} = \min_{\tau_i \in \mathcal{T}_{\text{HI}}} T_i.$$

Note that, we also require $\beta^N > 0$ and $\beta^C > 0$. Intuitively, it means that the maximum period of no available budget for the VP must be shorter than the shortest period of the supplied tasks; otherwise, there is no way to provide worst-case guarantees to the task with the shortest period.

Similar to many mixed-criticality scheduling algorithms, EDF-VDVP also has two modes to cope with the two estimations. In many prior work on multiple estimations on execution time, the mode switch point is rather straightforward — when a job has executed for its less pessimistic execution time without signaling completion. In our context, the less pessimistic budget Θ^N is the greater one, and therefore the mode switch cannot be in the same manner as before.

Mode switch. Under EDF-VDVP, there are two mode, namely the *nominal* mode and *critical* mode. The system always starts with the nominal mode. At every time instant, the scheduler keeps track of that 1) b time units budget have already received during the current resource period, and 2) there are still p time units until the end of current resource period. The mode switch must happen when a time unit has not supply available budget, *i.e.*, for some time instant t^*, the time unit $[t^*, t^* + 1)$ is not available to provide any execution for tasks in $\mathcal{T}$. If at time $t^* + 1$, it becomes true that $b + p < \Theta^N$, then EDF-VDVP is notice a mode switch to the critical mode. We define t^* (the time instant followed by an unavailable time unit) as the *mode-switch* time instant. Note that, the scheduler detecting

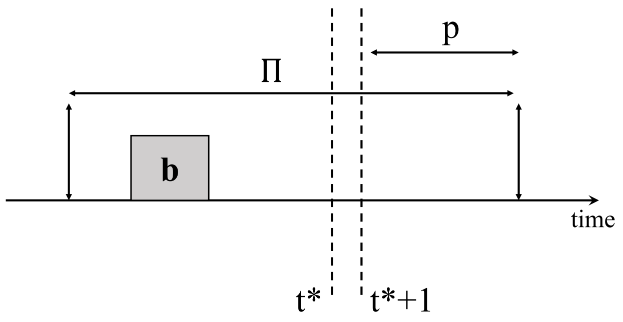

Fig. 3: An illustration for the mode-switch time instant t^*.

the mode switch at time instant t^* or t^*+1 does not affect any scheduling decision as $[t^*, t^*+1)$ is an unavailable time unit anyway. Fig. 3 illustrates the mode switch time instant t^*.

Scheduling with virtual deadlines. Besides the differences in the assumed system model, EDF-VDVP schedules tasks in the same virtual-deadline-based manner as EDF-VD does. Every job of HI-task τ_i has a virtual deadline $T_i' = x \cdot T_i$ time units after its release, where x is the system-wide virtual deadline scaling factor such that $0 < x \leq 1.0$. In contrast, the virtual deadlines of LO-tasks is the same as their actual deadlines. In the runtime, the EDF-VDVP always begins with nominal mode where the pending job with the earliest *virtual* deadline is scheduled for execution whenever available processing time is allocated for the VP. When a mode switch to the critical mode is triggered, EDF-VDVP discards any pending and upcoming LO-jobs, and afterwards, schedules the pending HI-job with earliest *actual* deadline for execution whenever available processing time is allocated for the VP.

Calculating the scaling factor x. For a given task set $\mathcal{T}$ and given parameters Π, Θ^N, the scaling fact x is calculated by

$$x = \frac{U_{\text{HI}}}{\beta^N - U_{\text{LO}}}. \tag{4}$$

By (1) and (2), $U_{\text{HI}} + U_{\text{LO}} = U \leq \beta^N$, *i.e.*, $U_{\text{HI}} \leq \beta^N - U_{\text{LO}}$. Therefore, it is clear that $0 < x \leq 1.0$.

IV. Schedulability Test

In this section, we present a sufficient schedulability test for EDF-VDVP. The schedulability test needs to validate the guarantees in the two modes corresponding to the nominal and critical budgets, respectively, so that the schedulability criteria defined in Sec. II are met.

The following lemma shows that with x being set by (4), the deadlines of all (HI- and LO-) tasks are met in the nominal mode, since the virtual deadline of any job is no later than its actual deadline.

Lemma 2. *All virtual deadlines of* HI- *and* LO-*tasks are met in the nominal mode if*

$$x \geq \frac{U_{\text{HI}}}{\beta^N - U_{\text{LO}}}.$$

Proof. In the nominal mode, treating the virtual deadlines as the actual deadlines, every HI-task τ_i can be viewed as a sporadic task τ_i' with a short period (and shorter relative deadline) $T_i' = x \cdot T_i$. Therefore,

$$u_i' = \frac{C_i}{T_i'} = \frac{C_i}{x \cdot T_i} = \frac{u_i}{x}.$$

On the other hand, the period, deadline, and therefore utilization of every LO-task remain unchanged. Furthermore, the budget supply in the nominal mode follows the periodic resource model with parameters (Π, Θ^N). Thus, by Theorem 1, the virtual deadlines of all tasks are met if

$$\sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \frac{u_i}{x} \leq \beta^N$$
$$\Leftrightarrow \quad U_{\text{LO}} + \frac{U_{\text{HI}}}{x} \leq \beta^N$$
$$\Leftrightarrow \quad x \geq \frac{U_{\text{HI}}}{\beta^N - U_{\text{LO}}}.$$

The last equivalence holds because (1) and (2) hold and the lemma follows. □

Furthermore, the following lemma provides a sufficient condition for all actual deadlines of HI-tasks being met in the critical mode.

Lemma 3. *All deadlines of* HI-*tasks in the critical mode will be met if*

$$x \leq 1 - \frac{U_{\text{HI}}}{\beta^C}.$$

Proof. At the mode switch point from the nominal to critical mode, a job from any task $\tau_i \in \mathcal{T}_{\text{HI}}$ must either be completed or have its virtual deadline after the mode switch point, because by Lemma 2, all virtual deadlines of HI-jobs are met in the nominal mode. That is, if not completed yet, a job of a HI-task τ_i must have an actual deadline at least $(1-x)T_i$ time units after this mode-switch point. Afterwards, any job from any task $\tau_i \in \tau_{\text{HI}}$ has at least $T_i > (1-x)T_i$ (as $0 < x < 1.0$) time units from their releases in the HI-mode to their corresponding deadlines.

That is, in the critical mode, every HI-task τ_i can be viewed as a sporadic task τ_i' with a short period (and shorter relative deadline) $T_i' = (1-x) \cdot T_i$. Therefore,

$$u_i' = \frac{C_i}{T_i'} = \frac{C_i}{(1-x) \cdot T_i} = \frac{u_i}{1-x}.$$

Furthermore, the budget supply in the critical mode follows the periodic resource model with parameters (Π, Θ^C). Thus, by Theorem 1, the actual deadlines of all HI-tasks are met in the critical mode if

$$\sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i' = \frac{U_{\text{HI}}}{1-x} \leq \beta^C.$$

Because $0 < x \leq 1.0$ and $\beta^C > 0$ as noted at the beginning of Sec. III, the lemma follows. $\square$

By Lemmas 2 and 3, the following theorem holds and serves as a sufficient schedulability test.

Theorem 2. *A mixed-criticality task set $\mathcal{T}$ is schedulable on a VP with resource period Π, nominal budget Θ^N, and critical budget Θ^C, if* (1), (2), *and* (3) *hold, and*

$$\frac{U_{\text{HI}}}{\beta^C} + \frac{U_{\text{HI}}}{\beta^N - U_{\text{LO}}} \leq 1. \tag{5}$$

V. Related work

In the last decade, multicore processors have become ubiquitous and there has been extensive research work on how to efficiently utilize these parallel machines with different types of real-time tasks, including mixed criticality real-time task scheduling and compositional real-time task scheduling.

The first work on the verification of a Mixed Criticality System was published by Vestal (of Honeywell Aerospace) in 2007 [41]. It used an extension of standard fixed priority (FP) real-time scheduling theory and proposed a restrictive work-flow model, focused on a single processor and made use of Response Time Analysis [3]. It showed that neither rate monotonic [30] nor deadline monotonic [29] priority assignment is optimal for MCS; however Audsleys optimal priority assignment algorithm [4] was found to be applicable. This paper was followed by two publications in 2008 by Baruah and Vestal [8], and Huber et al. [26].

Beside the basic mixed criticality task model, a bunch of techniques addressing mixed criticality systems in different scenarios are proposed: letting any LO-criticality job that has started, run to completion [9]; reducing the priorities of the LO-criticality tasks [7], or similar with EDF scheduling [25]; increasing the periods and deadlines of LO-criticality jobs [20] [27] [34] [37] [38] [39], called task stretching, the elastic task model or multi-rate; decreasing the computation times of some or all of the LO-criticality tasks [11], perhaps by utilising an imprecise mixed-criticality (IMC) model [24] or budget control [22]; moving some LO-criticality tasks to a different processor that has not experienced a criticality mode change [42]; improving resource utilization while guaranteeing safe execution of critical applications [21, 23, 28].

There also has been extensive research on compositional real-time scheduling. Insik Shin and Insup Lee present a formal description of compositional real-time scheduling problems in [35, 36]. They identify issues that need be addressed by solutions and provide their framework for the solutions, which is based on the periodic interface. Following this work, [16] introduces the Explicit Deadline Periodic (EDP) resource model, and present compositional analysis techniques under EDF and DM. It shows that these techniques are bandwidth optimal, in that they do not incur any bandwidth overhead in abstraction or composition. ARINC specification 653-2 [18] describes the interface between application software and underlying middle-ware in a distributed real-time avionics system. Authors develop compositional techniques for automated scheduling of partitions and processes in such systems. This work is followed by [12]. It proposes a compositional approach to formal specification and schedulability analysis of real-time applications running under a Time Division Multiplexing (TDM) global scheduler and preemptive Fixed Priority (FP) local schedulers, according to the ARINC-653 standard.

In addition to the traditional compositional real-time task scheduling, several novel scheduling algorithms[19][16][31] are proposed to schedule real-time task systems under different system architectures [43][33][15][10][44][32][13][2]. Correspondingly, to validate the schedulability of each real-time task system, schedulability analysis frameworks [16][17][1] are proposed for analyzing real-time tasks in different real-time applications.

VI. Conclusion

In this paper, we have made efforts to extend the fundamental ideas and frameworks of MC scheduling to a new dimension in the context of compositional real-time systems. We proposed to use multiple parameters to provision the VP supply. To further illustrate this approach, we formed up a specific scheduling problem in the context of a single virtual processor characterizing by the periodic resource model. We developed a virtual-deadline-based algorithm to solve it and presented a sufficient utilization-based schedulability test.

Future Work. The work in this paper can be further extended in several ways. First, in addition to implicit-deadline tasks, constrained- and arbitrary-deadline may be considered. Second, we plan to incorporate multiple budget estimations with the classic multiple WCET estimations together in MC scheduling. Also, in addition to the resource budget, multiple estimations on the resource period are another possible dimension and need further investigation. Finally, multiprocessor extension may need more efforts due to the complicated supply analysis induced by parallelism but is definitely an interesting topic.

Acknowledgment

This work is supported by start-up and REP grants from Texas State University and a start-up grant from Wayne State University.

References

[1] Madhukar Anand, Arvind Easwaran, Sebastian Fischmeister, and Insup Lee. Compositional feasibility analysis of conditional real-time task models. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 391–398. IEEE, 2008.

[2] Madhukar Anand, Sebastian Fischmeister, and Insup Lee. Composition techniques for tree communication schedules. In *19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, pages 235–246. IEEE, 2007.

[3] Neil Audsley, Alan Burns, Mike Richardson, Ken Tindell, and Andy J Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[4] Neil C Audsley, Alan Burns, Robert I Davis, Ken W Tindell, and Andy J Wellings. Fixed priority pre-emptive scheduling: An historical perspective. *Real-Time Systems*, 8(2-3):173–198, 1995.

[5] 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 *Proceedings of the 24th Euromicro Conference on Real-Time Systems*, pages 145–154, 2012.

[6] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In *Proceedings of the 19th Annual European Symposium on Algorithms*, pages 555–566, 2011.

[7] Sanjoy Baruah and Alan Burns. Implementing mixed criticality systems in ada. In *International Conference on Reliable Software Technologies*, pages 174–188. Springer, 2011.

[8] Sanjoy Baruah and Steve Vestal. Schedulability analysis of sporadic tasks with multiple criticality specifications. In *2008 Euromicro Conference on Real-Time Systems*, pages 147–155. IEEE, 2008.

[9] Sanjoy K Baruah, Alan Burns, and Robert I Davis. Response-time analysis for mixed criticality systems. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 34–43. IEEE, 2011.

[10] Abdeldjalil Boudjadar, Alexandre David, Jin Hyun Kim, Kim G Larsen, Marius Mikučionis, Ulrik Nyman, and Arne Skou. Hierarchical scheduling framework based on compositional analysis using uppaal. In *International Workshop on Formal Aspects of Component Software*, pages 61–78. Springer, 2013.

[11] Alan Burns and Sanjoy Baruah. Towards a more practical model for mixed criticality systems. In *Workshop on Mixed-Criticality Systems (colocated with RTSS)*, 2013.

[12] Laura Carnevali, Alessandro Pinzuti, and Enrico Vicario. Compositional verification for hierarchical scheduling of real-time systems. *IEEE Transactions on Software Engineering*, 39(5):638–657, 2012.

[13] Sanjian Chen, Linh TX Phan, Jaewoo Lee, Insup Lee, and Oleg Sokolsky. Removing abstraction overhead in the composition of hierarchical real-time systems. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 81–90. IEEE, 2011.

[14] Z. Deng and J. Liu. Scheduling real-time applications in an open environment. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, pages 308–319, 1997.

[15] Zheng Dong, Cong Liu, Soroush Bateni, Kuan-Hsun Chen, Jian-Jia Chen, Georg von der Brüggen, and Junjie Shi. Shared-resource-centric limited preemptive scheduling: A comprehensive study of suspension-based partitioning approaches. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 164–176. IEEE, 2018.

[16] Arvind Easwaran, Madhukar Anand, and Insup Lee. Compositional analysis framework using edp resource models. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 129–138. IEEE, 2007.

[17] Arvind Easwaran and Insup Lee. Compositional schedulability analysis for cyber-physical systems. *ACM Sigbed Review*, 5(1):6, 2008.

[18] Arvind Easwaran, Insup Lee, Oleg Sokolsky, and Steve Vestal. A compositional scheduling framework for digital avionics systems. In *2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTAS)*, pages 371–380. IEEE, 2009.

[19] Nathan Fisher and Farhana Dewan. Approximate bandwidth allocation for compositional real-time systems. In *2009 21st Euromicro Conference on Real-Time Systems*, pages 87–96. IEEE, 2009.

[20] Chris Gill, James Orr, and Steven Harris. Supporting graceful degradation through elasticity in mixedcriticality federated scheduling. In *Proc. 6th Workshop on Mixed Criticality Systems (WMC), RTSS*, pages 19–24, 2018.

[21] Xiaozhe Gu and Arvind Easwaran. Dynamic budget management with service guarantees for mixed-criticality systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 47–56. IEEE, 2016.

[22] Xiaozhe Gu and Arvind Easwaran. Dynamic budget management and budget reclamation for mixed-criticality systems. *Real-Time Systems*, pages 1–46, 2019.

[23] Xiaozhe Gu, Arvind Easwaran, Kieu-My Phan, and Insik Shin. Resource efficient isolation mechanisms in mixed-criticality scheduling. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 13–24. IEEE, 2015.

[24] Lin Huang, I Hou, Sachin S Sapatnekar, Jiang Hu, et al. Graceful degradation of low-criticality tasks in multiprocessor dual-criticality systems. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, pages 159–169. ACM, 2018.

[25] Pengcheng Huang, Georgia Giannopoulou, Nikolay Stoimenov, and Lothar Thiele. Service adaptions for mixed-criticality systems. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 125–130. IEEE, 2014.

[26] Bernhard Huber, Christian El Salloum, and Roman Obermaisser. A resource management framework for mixed-criticality embedded systems. In *2008 34th Annual Conference of IEEE Industrial Electronics*, pages 2425–2431. IEEE, 2008.

[27] Mathieu Jan, Lilia Zaourar, and Maurice Pitel. Maximizing the execution rate of low criticality tasks in mixed criticality system. *Proc. WMC, RTSS*, pages 43–48, 2013.

[28] Jaewoo Lee, Kieu-My Phan, Xiaozhe Gu, Jiyeon Lee, Arvind Easwaran, Insik Shin, and Insup Lee. Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors. In *2014 IEEE Real-Time Systems Symposium*, pages 41–52. IEEE, 2014.

[29] Joseph Y-T Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4):237–250, 1982.

[30] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[31] Shanmuga Priya Marimuthu and Samarjit Chakraborty Chakraborty. A framework for compositional and hierarchical real-time scheduling. In *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, pages 91–96. IEEE, 2006.

[32] Andrew Nelson, Kees Goossens, and Benny Akesson. Dataflow formalisation of real-time streaming applications on a composable and predictable multi-processor soc. *Journal of Systems Architecture*, 61(9):435–448, 2015.

[33] Linh TX Phan, Meng Xu, Jaewoo Lee, Insup Lee, and Oleg Sokolsky. Overhead-aware compositional analysis of real-time systems. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 237–246. IEEE, 2013.

[34] Saravanan Ramanathan, Arvind Easwaran, and Hyeonjoong Cho. Multi-rate fluid scheduling of mixed-criticality systems on multiprocessors. *Real-Time Systems*, 54(2):247–277, 2018.

[35] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pages 1–12, 2003.

[36] Insik Shin and Insup Lee. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):30, 2008.

[37] Hang Su, Peng Deng, Dakai Zhu, and Qi Zhu. Fixed-priority dual-rate mixed-criticality systems: Schedulability analysis and performance optimization. In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 59–68. IEEE, 2016.

[38] Hang Su, Nan Guan, and Dakai Zhu. Service guarantee exploration for mixed-criticality systems. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–10. IEEE, 2014.

[39] Hang Su and Dakai Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 147–152. IEEE, 2013.

[40] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE Real-Time Systems Symposium*, pages 239–243, 2007.

[41] Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*.

[42] Hao Xu and Alan Burns. Semi-partitioned model for dual-core mixed criticality system. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, pages 257–266. ACM, 2015.

[43] Meng Xu, Linh Thi Xuan Phan, Oleg Sokolsky, Sisu Xi, Chenyang Lu, Christopher Gill, and Insup Lee. Cache-aware compositional analysis of real-time multicore virtualization platforms. *Real-Time Systems*, 51(6):675–723, 2015.

[44] Jungwoo Yang, Hyungseok Kim, Sangwon Park, Changki Hong, and Insik Shin. Implementation of compositional scheduling framework on virtualization. *ACM SIGBED Review*, 8(1):30–37, 2011.