Compositional Mixed-Criticality Systems with Multiple Executions and Resource-Budgets Model

Abdullah Al Arafat*, Sudharsan Vaidhun†, Liangkai Liu‡, Kecheng Yang§, Zhishan Guo*
*North Carolina State University, †University of Central Florida, ‡Wayne State University, §Texas State University
{aalaraf, zguo32}@ncsu.edu, sudharsan.vaidhun@knights.ucf.edu, liangkai@wayne.edu, yangk@txstate.edu

***Abstract*—Software reusability and system modularity are key features of modern autonomous systems. As a consequence, there is a rapid shift towards hierarchical and compositional architecture, as evidenced by AUTOSAR in automobiles and ROS2 in robotics. The resource-budget supply model is widely applied in the real-time analysis of such systems. Meanwhile, real-time systems with multiple critical levels have received significant attention from the research community and industry. These systems are designed with multiple execution budgets for multiple system-critical levels. Existing studies on mixed-criticality systems consider a dedicated resource supply. This paper considers a novel generalized system model with multiple execution estimations and resource-budget supplies for compositional systems. An analytical model and a demand-bound function-based schedulability test are presented for the EDF-based scheduler in the proposed compositional mixed-criticality system. A range for setting the resource supply period is derived to ensure the schedulability of workloads when supply budgets are known. The general performance of the scheduling framework and its wider applicability is further demonstrated and evaluated using synthetic workloads and resource models, where synthetic workload parameters are derived through a case study on an autonomous driving system.**

***Index Terms*—Mixed-Criticality, Compositional Scheduling, Resource Criticality, Demand Bound Functions, AUTOSAR**

I. Introduction

Autonomous systems, such as autonomous vehicles, industrial robots, drones, *etc.*, are the next major computing demand drivers. Most of these systems are safety-critical, and their *functional* and *temporal* correctness must be *verifiable*. Due to the sophisticated design and underlying hardware-platform complexities, software reusability and system modularity are required design choices for these systems. An embedded system design paradigm has emerged to support the software reusability and system modularity of these systems, requiring an additional software layer between the function and hardware layers, such as those defined in AUTOSAR [3] and ROS2 [2]. The design of such an additional software layer makes a rapid shift towards compositional (also known as hierarchical) architecture, as evidenced by, *e.g.*, AUTOSAR [3], [17] in automobiles and ROS2 [2] in robotics.

Compositional scheduling framework has emerged as an efficient way to provide temporal isolation among applications and has been used as one of the common scheduling frameworks for mixed-criticality (MC) systems in practice [42]. Typically, compositional scheduling framework maintains two layers of schedulers: *local scheduler* schedules the workloads in a virtual processor (VP) and *global scheduler* schedules all the VP's providing resource supply from the underlying hardware platform [43]. Scheduling workloads of an application in a VP ensures the modularity and reusability of the application.

Usually, the resource supply model consists of a single supply budget for each VP [43]. However, in scheduling MC workloads [47] in a VP, a single resource supply results in sub-optimal performance [28]. Therefore, Lackorzyński *et al.* [28] developed a multiple-supply-based scheduling scheme to schedule MC workloads in a VP. Later, Gu *et al.* [24] also proposed a dual-supply-based hierarchical scheduling framework for resource efficient scheduling of MC workloads. However, these methods either relaxed the isolation constraints of VPs [28] or added awareness of workload criticality-based mode-switch of local scheduler to global scheduler [24]. Although these relaxations significantly reduce the local scheduler's scheduling complexity (*e.g.*, schedulable 'system modes' are restricted to only the modes introduced by workload criticalities), it may break some of the critical properties of compositional scheduling framework, such as temporal isolation, modularity, and reusability.

We, instead, consider a dual-critical resource model for MC workload scheduling in a standard hierarchical scheduling framework, where the local and global schedulers are *independent* [43]. Specifically, our resource supply model is driven by a *practical* one designed for AUTOSAR [3], which consists of a 'nominal' and 'critical' supply budget. Such a resource supply model is currently available on adaptive partition scheduling (APS) in QNX [17]. Notably, APS can be used in ROS2 too [17]. To illustrate the necessity of this resource model, let us consider a scenario where resource supply is not consistent due to critical resource requirements from some VPs in the system. To provide such critical supplies, other VPs will receive a degraded supply. In such case, it is important to guarantee that the VP with degraded resource supply should at least schedule high-critical workloads. To tackle such a scenario, we consider a dual-critical resource supply with a 'nominal' resource supply during regular system operations and a 'critical' supply where resource supply is lower than normal due to critical system operations. Notably, however, VPs with 'surplus' resource supply due to critical resource requirements are expected to complete critical workloads of those VPs. Hence, we do not consider to tackle issues related to 'surplus' resource supply.

Recently, Yang and Dong [50] proposed an MC model for resource budgets in compositional scheduling. The authors

considered a dual-critical resource supply model for a VP in a hierarchical scheduling scheme in the proposed MC model. However, the paper did not consider the workload criticality model presented in Vestal's MC model [47] to mitigate the curse of pessimism induced by the overly-pessimistic worst-case-execution-time (WCET) estimations of workloads. In contrast, we have developed scheduling strategies for MC workloads with multiple execution times in a VP with dual-critical resource supply. Notice that scheduling MC workloads with multiple execution times on a dual-criticality resource supply introduces new scheduling challenges, which is the focus of this paper. It is mainly caused by the necessity of *independently system mode-switch triggering for workloads and resource supplies* (details presented in Sec. II-C).

Contributions. In this paper, we present *MC-Budget*, the first general MC model that includes both resource and workload criticalities independently for a VP under the compositional scheduling framework. As both MC and compositional systems become jointly employed and more popular in safety-critical systems, MC-Budget will be a critical foundation for real-time analysis and verification purposes. Specifically, we consider a dual-critical workload model where each task is either a low- or high-critical task with two estimations (*e.g.*, optimistic and pessimistic) of execution time. Regarding resource supply to a VP, we consider a 'nominal' and a 'critical' resource supply. Overall, MC-Budget consists of four system modes, each with a combination of resource and workload criticality levels. In particular, the technical contributions to this paper are summarized as follows:

- We propose the first generalized mixed-criticality system model (MC-Budget), which subsumes the state-of-the-art (*e.g.*, [47], [50]) as special cases under the MC scheduling framework.
- We present a new demand bound function-based *pseudo-polynomial* schedulability test for the EDF-based scheduler of the proposed MC model.
- We derive a range for the resource period, ensuring a given workload's schedulability.
- We conduct extensive schedulability evaluations using synthetic workloads and resource models, where the workload parameters are estimated through the simulation of a fully functional autonomous driving system.

Organization. The remaining of the paper is organized as follows: Section II introduces the detailed system model, its background and then proposes a scheduler that handles such systems. Section III and IV present a corresponding schedulability test and a bound for setting the resource supply period, respectively. Section V illustrates an algorithm to jointly determine hyper-parameters (*e.g.*, deadline shrinkage parameter and resource period) for the schedulable workload. Section VI demonstrates a detailed evaluation of the scheduler and the associated schedulability test using synthetic workloads. Section VII discusses the related work. Finally, Section VIII provides the concluding remarks of the paper.

II. System Model and Background

The scheduling model consists of a workload model that describes the tasks and a resource model that describes the available resources, and scheduling strategies (algorithm) to schedule the workloads on the available resources. This section formally describes the workload and resource model for a set of constrained-deadline sporadic tasks to be scheduled on a virtual processor. We then present the scheduling strategy of the proposed scheduling problem.

A. Workload Model

Let us consider a constrained-deadline sporadic task set of k independent and preemptive MC tasks, $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_k\}$. Each task τ_i consists of a five-tuple, $\tau_i = (C_i^{\text{LO}}, C_i^{\text{HI}}, T_i, D_i, \chi_i)$ that can release (potentially) infinite sequence of instances with a minimum inter-arrival separation of T_i and the relative deadline of the task for each instance is $D_i (\leq T_i)$ time units. We consider two estimations of execution time for each task τ_i—$C_i^{\text{LO}}, C_i^{\text{HI}}$ which are the optimistic execution time and worst-case-execution-time (WCET) of τ_i, respectively. The parameter $\chi_i \in \{\text{LO}, \text{HI}\}$ represents the criticality of task τ_i. Each task in the system is either a low or high critical task. For instance, if $C_i^{\text{LO}} < C_i^{\text{HI}}$, then the task is a high-critical task ($\chi_i = \text{HI}$) and a system mode switch can be initiated in case of over execution of C_i^{LO}. In contrast, if $C_i^{\text{LO}} = C_i^{\text{HI}}$, then the task is a low-critical task ($\chi_i = \text{LO}$) and there would not be a system mode switch for overrun of C_i^{LO}. For brevity, we will term the low- and high-critical tasks as LO-tasks and HI-tasks and define LO- and HI-task set as $\Gamma_{\text{LO}} = \{\tau_i \in \Gamma | \chi_i = \text{LO}\}$ and $\Gamma_{\text{HI}} = \{\tau_i \in \Gamma | \chi_i = \text{HI}\}$, respectively in rest of the paper.

Utilization of a task τ_i is defined as the ratio of execution time and period of the task, $u_i(p) = C_i^p / T_i$, where $p \in \{\text{LO}, \text{HI}\}$. For example, $u_i(\text{LO})$ denotes the utilization of task τ_i using optimistic execution time, C_i^{LO}. We define task set utilization as follows,

$$U_p^{p'} = \sum_{\tau_i \in \Gamma \,|\, \chi_i = p' \in \{\text{LO}, \text{HI}\}} u_i(p) \tag{1}$$

For instance $U_{\text{LO}}^{\text{HI}}$ is the utilization of HI-task set with optimistic execution times, C_i^{LO}'s.

Demand bound function (DBF), $\text{dbf}(\tau_i, \ell)$**,** gives an upper bound of maximum possible execution of all jobs of task, $\tau_i = (T_i, C_i, D_i)$ that have both their arrival times and deadlines in the scheduling window, ℓ. The demand bound function is defined as follows [11],

$$\text{dbf}(\tau_i, \ell) = \left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \cdot C_i \tag{2}$$

Note, $\forall \ell \geq 0$, $\text{dbf}(\tau_i, \ell) \geq 0$, as $\left\lfloor \frac{D_i}{T_i} \right\rfloor + 1 \geq 0 | D_i \leq T_i$.

Fact 1. *A linear demand bound function of task* τ_i, $\text{ldbf}(\tau_i, \ell)$ *is,*

$$\text{ldbf}(\tau_i, \ell) = \begin{cases} 0 & \text{if } \ell \leq D_i \\ \frac{\ell - D_i}{T_i} + 1 \cdot C_i & \text{if } \ell > D_i \end{cases} \tag{3}$$

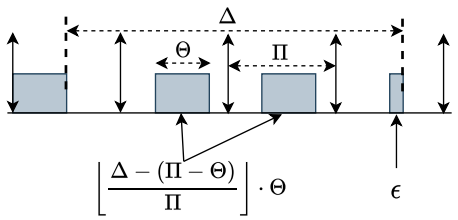


Fig. 1: Worst-case resource supply of a periodic resource model $\mathcal{R}(\ ;\ )$ [43].

where $\mathrm{ldbf}(\ _i;\grave{}) \geq \mathrm{dbf}(\ _i;\grave{}) \ \forall \grave{} \geq 0$.

B. Resource Model

We consider a periodic partitioned resource model $\mathcal{R}(\ ;\ )$, where a virtual processor (VP) periodically receives a certain amount of resource budget $= \{\ ^n;\ ^c\}$ in between a specified time period . Depending on the resource criticality, the resource budget is either a nominal budget ^{n}, or a critical budget ^{c} where $^n > \ ^c$. Let us define nominal resource model as $\mathcal{R}_n(\ ^n;\ )$ and critical resource model as $\mathcal{R}_c(\ ^c;\ )$. We further define *nominal* and *critical bandwidth* of resource as,

$$!^n = \frac{^n}{} \quad \text{and} \quad !^c = \frac{^c}{};\ \text{respectively.}$$

Supply bound function, sbf, is the *minimum* resource supply during an interval of . The $\mathrm{sbf}(\ )$ of a periodic partition resource model $\mathcal{R}(\ ;\ )$ is given by [43] as,

$$\mathrm{sbf}_R(\ ) = \begin{cases} 0; & \text{if} \quad \leq 2\cdot(\ -\ ) \\ \underline{(\quad)}\cdot\ +\ ; & \text{otherwise} \end{cases} \tag{4}$$

where,

$$= \max\ -2(\ -\ ) - \frac{-(\ -\ )}{};0$$

Figure 1 illustrates the $\mathrm{sbf}_R(\ )$ considering a worst-case scenario to show minimum resource supply during an interval. Note that maximum non-supply interval is zero and the sbf() is a non-decreasing function.

Fact 2. *(Theorem 1 [43]) A constrained-deadline task set, , with a periodic resource model $\mathcal{R}(\ ;\ )$ can be successfully scheduled by a dedicated resource supply,* if and only if

$$\sum_{_i 2} \mathrm{dbf}(\ _i;\grave{}) \leq \mathrm{sbf}_R(\grave{});\forall \grave{} \geq 0: \tag{5}$$

Fact 3. *(Lemma 1 [43]) The linear supply bound function* $\mathrm{lsbf}(\ )$ *of* $\mathcal{R}(\ ;\ )$ *is,*

$$\mathrm{lsbf}_R(\ ) = —(\ -2\cdot(\ -\ )) \leq \mathrm{sbf}_R(\ )$$

So, the *nominal* $\mathrm{lsbf}_{R_n}(\ )$ of $\mathcal{R}_n(\ ^n;\ )$ and *critical* $\mathrm{lsbf}_{R_c}(\ )$ of $\mathcal{R}_c(\ ^c;\ )$ are,

$$\mathrm{lsbf}_{R_n}(\ ) = !^n\cdot(\ -2\cdot(\ -\ ^n)) \tag{6}$$

$$\mathrm{lsbf}_{R_c}(\ ) = !^c\cdot(\ -2\cdot(\ -\ ^c)) \tag{7}$$

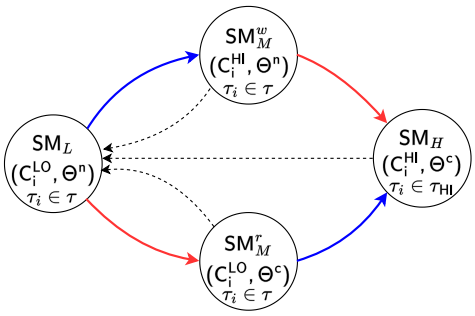


Fig. 2: Proposed state diagram of system modes. Mode switches are indicated by solid lines (blue and red lines indicated that mode switches are initiated by HI-*task's overrun* and *scarcity of resource supply*, respectively). Dotted lines indicate the mode restoration direction. Note that some jobs of each task $_i \in$ LO in system modes SM_M^w and SM_M^r are dropped following graceful degradation strategy, while all jobs of HI-tasks are executed by its deadline in all system modes.

Now we define pseudo-inverse of $\mathrm{sbf}(\ )$ as follows,

$$\overline{\mathrm{sbf}}(\ ) = \min\{\ \mid \mathrm{sbf}(\ ) = \ \} \tag{8}$$

$\overline{\mathrm{sbf}}(\ )$ provides the minimum time to serve a resource .

The pseudo-inverse of $\mathrm{lsbf}_{R_n}(\ )$ and $\mathrm{lsbf}_{R_c}(\ )$ are,

$$\overline{\mathrm{lsbf}_{R_n}}(\ ) = \frac{1}{!^n}\cdot\ +2\cdot(\ -\ ^n) \tag{9}$$

$$\overline{\mathrm{lsbf}_{R_c}}(\ ) = \frac{1}{!^c}\cdot\ +2\cdot(\ -\ ^c) \tag{10}$$

Equations 9 and 10 are directly derived from the linear Equations 6 and 7, respectively.

Fact 4. *(derived from Fact 3) The pseudo-inverse of* $\mathrm{lsbf}(\ )$ *is always greater or equal than the pseudo-inverse of* $\mathrm{sbf}(\ )$,

$$\overline{\mathrm{lsbf}}(\ ) \geq \overline{\mathrm{sbf}}(\ ) \tag{11}$$

C. System Modes

We consider four system modes based on the criticality of workloads and resource budgets. Figure 2 presents a state diagram of system states illustrating the possible system mode switching scenarios. We define the system modes as follows:

System Mode–*Low* ($\mathrm{SM_L}$): All workloads are executed up to C_i^{LO} with a resource supply budget of ^{n}.

System Mode–*Medium* ($\mathrm{SM_M}$): This system mode can be triggered by either an overrun of workload or scarcity of resources. So, we further define two distinct system modes for $\mathrm{SM_M}$.

- $\mathrm{SM_M}$ with ^{c} ($\mathrm{SM_M^r}$): In this mode, all workloads are executed up to its C_i^{LO} with a resource budget of ^{c}.
- $\mathrm{SM_M}$ with $C_i^{\mathrm{HI}} \mid\ _i \in$ HI ($\mathrm{SM_M^w}$): In this mode, all tasks executes up to C_i^{HI} (Note, $C_i^{\mathrm{HI}} = C_i^{\mathrm{LO}} \mid\ _i \in$ LO) with a resource supply budget of ^{n}.

System Mode–*High* ($\mathrm{SM_H}$): In $\mathrm{SM_H}$, only HI-tasks execute up to its WCET, C_i^{HI} with a resource budget of ^{c}.

Note that once a criticality is raised (either for a task overrun or resource scarcity) and system mode is switched

accordingly, the system cannot return to the previous system mode in the next clock tick. This property is certainly held as a higher criticality mode is needed to address specific system requirements for correct operation. In contrast, the system can switch to a higher critical mode to address more criticalities in the subsequent clock tick. Now, however, suppose both resource and task overrun criticalities trigger simultaneously. In that case, we address the criticalities in sequential order, such as any one of the criticality triggers the first system mode switch to one of SM_M. The other criticality then triggers the system mode switch to SM_H respectively. All other valid system mode transitions are illustrated in Fig. 2.

Algorithm 1: Acceptance Test of LO-tasks in SM_M Mode [25]

Input: Acceptance ratios of LO-tasks, $\{r_i\}$

```
for ∃ τ_i ∈ τ_LO do
    a_i ← 0 ;        // Initialize acceptance job counter
end
while TRUE do
    //Check whether p^th job J_p of τ_i is released;
    if J_p ∈ τ_i|χ_i = LO release then
        if a_i ≤ r_i · p then
            //Accept the job to schedule;
            a_i ← a_i + 1 ;        // Increase counter
        end
        else
            //Drop the job;
        end
    end
end
```

Graceful Degradation of LO-tasks in SM_M. We consider a graceful degradation of LO-tasks in SM^r_M and SM^w_M modes, where some jobs of each LO-task are dropped, and others are executed up to its WCET (C^{LO}_i). For LO-job admission/drop in SM^r_M and SM^w_M, following the existing work on graceful mixed-criticality scheduling [25], we maintain a fixed ratio, r_i where $\lceil r_i \cdot x \rceil$ jobs of every x jobs that released in SM_M mode are accepted for execution. The LO-jobs acceptance test in SM^r_M and SM^w_M modes is presented in Algorithm 1 (adopted from [25]). Algorithm 1 keeps accepting jobs until the number of accepted jobs becomes equal to the product of the ratio times released jobs counter. Note that this ratio holds for any number of released jobs. To include the acceptance ratio, r_i in each LO-task, we modify the LO-task tuple as: $\tau_i = (C^{\text{LO}}_i; T_i; D_i; r_i; \chi_i = \text{LO})$. We ignored C^{HI}_i here due to the fact that, for LO-tasks, $C^{\text{LO}}_i = C^{\text{HI}}_i$.

Remark 1. *Note that we consider dropping all active* LO*-jobs when a mode switch instant triggered from* SM_L *to* SM^r_M *or* SM^w_M*. As we drop only a single instance of* LO*-tasks during the mode-switch instant, it will not affect the performance of* LO*-tasks in the system. Hence, all first released instant of* LO*-tasks at* SM^r_M *and* SM^w_M *are accepted to execute following the acceptance test in Algorithm 1.*

Mode Switches. Each virtual processor (VP) starts with SM_L.

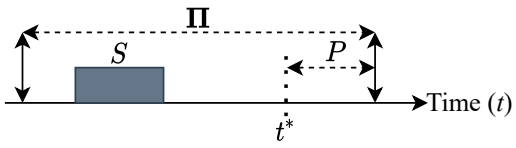


Fig. 3: Illustration of resource scarcity instant. In a supply period , suppose a VP received supply S. At the earliest time instant t , the remaining period of resource supply for the VP is P such that $S + P <$ ^{n}. Then the resource scarcity is triggered at t .

System switches to SM^w_M or SM^r_M from SM_L due to either overrun of C^{LO}_i by any HI-task or the scarcity of regular resource budget, ^{n}, respectively. The system triggers a mode switch from SM^w_M mode to SM_H if there is resource scarcity of regular supply budget ^{n} in the system and changes the supply budget from ^{n} to ^{c}. Similarly, a mode switch from SM^r_M to SM_H is initiated if a HI-task overruns its execution budget C^{LO}_i. As illustrated in Fig. 2, the system triggers a mode switch (*e.g.*, valid mode switch denoted by solid lines) following the rules mentioned above. In case of the mode switch in *reverse direction* as illustrated in Fig. 2 by the dotted lines, we consider the system can only restore to the SM_L from any other system modes when the current system mode remain idle at or after the resource replenishment instant. Note that it might be possible to switch back earlier than the idle instant of resource replenishment instant; however, for simplicity, we consider absolute idle instant (*e.g.*, system restart) for restoring to the initial system mode.

Remark 2. *Note that tracking whether a VP could receive a* nominal budget *or not is different from tracking whether a task has overrun. In the case of tracking task overruns, we only need to measure the execution time of the task and compare it with its WCET. Typically, a single counter (e.g., Watchdog) monitors task overrun issues. However, in the case of resource budget tracking, we need to track both the received budget and the available remaining resource period. Therefore, we should use two counters – the first to measure the unconsumed budget, and the second to measure the budget available within the resource period for consumption. The system will raise the resource scarcity flag whenever the remaining budget exceeds the remaining budget period. Figure 3 illustrates an instance of a mode-switch trigger due to resource scarcity.*

D. Scheduling Algorithm

Earliest-deadline-first (EDF) algorithm is an optimal scheduling algorithm for non-mixed-criticality workload scheduled on a dedicated supply uniprocessor platform [32]. Baruah *et al.* [12] developed a modified version of EDF with virtual deadlines for HI-tasks (EDF-VD) for mixed-criticality workload scheduling on a uniprocessor platform. We briefly describe the EDF-VD for MC systems [12] as follow.

EDF-VD [12]. In EDF-VD scheduling of a dual-critical regular MC system, HI-tasks are scheduled following a virtual deadline (typically, shorter than the actual deadlines),

$D_i^v = x \cdot D_i$ for $0 < x \leq 1$ and LO-tasks are scheduled by their actual deadline in low-system-critical mode. The parameter x is a deadline *'shrinkage factor'* for HI-tasks. All LO-tasks are dropped immediately in high-system-mode, and HI-tasks are scheduled following their actual deadlines.

However, the scheduling problem we present in this paper is entirely different from the existing problems addressed by EDF-VD. For the system mode-switches, a mode switch occurs only when at least one of two conditions is met – resource get scarce or a HI-task overruns. In addition, we allow graceful degradation of LO-tasks execution in SM_M^w and SM_M^r modes. We derive virtual deadlines of HI-tasks using Algorithm 2.

Remark 3. *Note that in regular* MC systems*, EDF-VD leverages the advantages of virtual deadlines of* HI*-tasks to fulfill the additional execution demands (e.g., $C_i^{\text{HI}} - C_i^{\text{LO}}$) for carry-over jobs*[1] *after a mode switch instant, which is triggered only by overrun of* HI*-tasks. In addition to overrun of* HI*-tasks, our system model also supports mode switches (e.g.,* SM_L *to* SM_M^r *and* SM_M^w *to* SM_H*) triggered by resource scarcity. Unlike regular MC systems, after a resource scarcity-triggered mode switch, our algorithm maintains the same deadline parameters (e.g., D_i or D_i^v) in the new system mode.*

Scheduling Strategies. The scheduling strategies for each system mode are as follows:

In SM_L, all workloads are executed following EDF up to its WCET, C_i^{LO}. Each LO-task must complete its execution within its deadline D_i. Each HI-task must complete execution by its virtual deadline, D_i^v.

In SM_M^r (or, SM_M^w), for any LO-task, τ_i, only $\lceil r_i \cdot x \rceil$ out of each x jobs are required to be executed up to its C_i^{LO} on or before its deadline D_i (Algorithm 1). In case of HI-tasks, all tasks executed either up to C_i^{LO} in SM_M^r mode on or before its virtual deadline, D_i^v or C_i^{HI} in SM_M^w mode by their actual deadline, D_i, respectively.

In SM_H, all LO-tasks are dropped immediately. HI-tasks execute up to its C_i^{HI} on or before its deadline, D_i.

Remark 4. *Although our focus in this paper is the schedulability analysis of the local scheduler for a VP, having a valid schedulability guarantee for the global scheduler is necessary to ensure end-to-end schedulability of a hierarchical and compositional scheduling framework. When it comes to global scheduling, a common strategy is to model each VP as an independent task to abstract resource demand by the VP, and then the global scheduler ensures a schedulability guarantee by providing sufficient resource supply to the VP [43]. We assume that the global scheduler is guaranteed to be schedulable with a 'nominal' and 'critical' resource budget model for each VP. Note that, due to the dual-resource supply model provided by the global scheduler, all valid mode-switches illustrated in Fig. 2 are feasible for dual-critical workloads.*

[1] A carry-over job is a job that released and remained unfinished before a mode-switch instant.

III. Schedulability Test

The schedulability analysis for EDF-VD is usually performed either using utilization-based analysis [6], or DBF-based analysis [20], [18]. Utilization-based tests can be performed in linear time complexity; however, the tests are sufficient only, and involves huge pessimism for constrained deadline systems (as the test becomes density-based). In this paper, we will develop a DBF-based test with pseudo-polynomial time complexity.

Ekberg and Yi [20] developed DBF-based analysis for regular MC systems, where they analyzed the DBF-based schedulability for each mode separately. Before deriving the DBFs for different system modes, we define the following functions for DBFs of LO- and HI-tasks in each mode:

$\text{dbf}_{\text{LO}}^{\text{SM}_\text{L}}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{LO}}$ in SM_L mode for any time interval of length ℓ.

$\text{dbf}_{\text{HI}}^{\text{SM}_\text{L}}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{HI}}$ in SM_L mode for any time interval of length ℓ.

$\text{dbf}_{\text{LO}}^{\text{SM}_\text{M}}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{LO}}$ in SM_M^r mode for any time interval of length ℓ.

$\text{dbf}_{\text{HI}}^{\text{SM}_\text{M}^r}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{HI}}$ in SM_M^r mode for any time interval of length ℓ.

$\text{dbf}_{\text{LO}}^{\text{SM}_\text{M}^w}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{LO}}$ in SM_M^w mode for any time interval of length ℓ.

$\text{dbf}_{\text{HI}}^{\text{SM}_\text{M}^w}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{HI}}$ in SM_M^w mode for any time interval of length ℓ.

$\text{dbf}_{\text{HI}}^{\text{SM}_\text{H}}(\tau_i, \ell)$—the demand of a task $\tau_i \in \tau_{\text{HI}}$ in SM_H mode for any time interval of length ℓ.

Now we use the Fact 2 to derive the DBF-based schedulability test of our proposed system.

Theorem 1. *A mixed-criticality workload τ can be successfully scheduled on* MC-Budget *system with a resource supply $\mathcal{R}(\Pi, \Theta)$ if each of the following conditions is satisfied for $\forall \ell \geq 0$:*

$$(A): \sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}_{\text{LO}}^{\text{SM}_\text{L}}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{HI}}^{\text{SM}_\text{L}}(\tau_i, \ell) \leq \text{sbf}_{R_n}(\ell)$$

$$(B): \sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}_{\text{LO}}^{\text{SM}_\text{M}^w}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{HI}}^{\text{SM}_\text{M}^w}(\tau_i, \ell) \leq \text{sbf}_{R_n}(\ell)$$

$$(C): \sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}_{\text{LO}}^{\text{SM}_\text{M}^r}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{HI}}^{\text{SM}_\text{M}^r}(\tau_i, \ell) \leq \text{sbf}_{R_c}(\ell)$$

$$(D): \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{HI}}^{\text{SM}_\text{H}}(\tau_i, \ell) \leq \text{sbf}_{R_c}(\ell)$$

Theorem 1 can be directly employed up on the calculation of DBFs and SBF for each condition. We can directly compute the SBF of each condition using Equation 4. Next, we will compute the DBFs.

Compute $\text{dbf}_{\text{LO}}^{\text{SM}_\text{L}}(\tau_i, \ell)$ and $\text{dbf}_{\text{HI}}^{\text{SM}_\text{L}}(\tau_i, \ell)$. In SM_L mode, all LO-tasks are executed by its deadline and HI-tasks are executed

by its virtual deadline. Therefore, the DBFs of SM_L can be directly calculated following Equation 2,

$$\forall\ _i \in\ _{\text{LO}};\ \text{dbf}^{\text{SM}_\text{L}}_{\text{LO}}(\ _i; `) = \left\lfloor \frac{` - D_i}{T_i} \right\rfloor + 1 \cdot C^{\text{LO}}_i$$

$$\forall\ _i \in\ _{\text{HI}};\ \text{dbf}^{\text{SM}_\text{L}}_{\text{HI}}(\ _i; `) = \left\lfloor \frac{` - D^v_i}{T_i} \right\rfloor + 1 \cdot C^{\text{LO}}_i$$

Compute $\text{dbf}^{\text{SM}^w_\text{M}}_{\text{LO}}(\ _i; `)$. As we drop any carry-over job from LO-tasks, computation $\text{dbf}^{\text{SM}^w_\text{M}}_{\text{LO}}(\ _i; `)$ is straightforward. However, we consider graceful degradation of LO-tasks. Therefore, we need to consider only $\lceil r_i \cdot x \rceil$ job's out of x consecutive releases while computing the DBF of $\forall\ _i \in\ _{\text{LO}}$ as follows,

$$\text{dbf}^{\text{SM}^w_\text{M}}_{\text{LO}}(\ _i; `) = \left\lceil r_i \cdot \frac{` - D_i}{T_i} \right\rceil + 1 \cdot C^{\text{LO}}_i \quad (12)$$

Compute $\text{dbf}^{\text{SM}^w_\text{M}}_{\text{HI}}(\ _i; `)$. The DBF of HI-tasks in $\text{SM}^w_\text{M}$ can be computed following the approach presented in [20]. Let us consider $\text{full}(\ _i; `)$ as the maximum demand of a HI-task including any potential carry-over job's in SM^w_M and $\text{done}(\ _i; `)$ as the minimum demand of a carry-over HI-task that must complete in $\text{SM}_\text{L}$. Then the $\text{dbf}^{\text{SM}^w_\text{M}}_{\text{HI}}(\ _i; `)$ is computed as,

$$\forall\ _i \in\ _{\text{HI}};\ \text{dbf}^{\text{SM}^w_\text{M}}_{\text{HI}}(\ _i; `) = \text{full}(\ _i; `) - \text{done}(\ _i; `) \quad (13)$$

$\text{full}(\ _i; `)$ and $\text{done}(\ _i; `)$ can be computed as follows [20],

$$\text{full}(\ _i; `) = \left\lfloor \frac{` - (D_i - D^v_i)}{T_i} \right\rfloor + 1 \cdot C^{\text{HI}}_i \quad (14)$$

$$\text{done}(\ _i; `) = \begin{cases} \max\{C^{\text{LO}}_i - m + D_i - D^v_i; 0\}; & \text{if } D_i - D^v_i \le m \le D_i \\ 0; & \text{otherwise} \end{cases} \quad (15)$$

where, $_i \in\ _{\text{HI}}$ and $m = ` \bmod T_i$.

Compute $\text{dbf}^{\text{SM}^r_\text{M}}_{\text{LO}}(\ _i; `)$. Computation $\text{dbf}^{\text{SM}^r_\text{M}}_{\text{LO}}(\ _i; `)$ is similar to $\text{dbf}^{\text{SM}^w_\text{M}}_{\text{LO}}(\ _i; `)$ and computed as follows for $\forall\ _i \in\ _{\text{LO}}$,

$$\text{dbf}^{\text{SM}^r_\text{M}}_{\text{LO}}(\ _i; `) = \left\lceil r_i \cdot \frac{` - D_i}{T_i} \right\rceil + 1 \cdot C^{\text{LO}}_i$$

Compute $\text{dbf}^{\text{SM}^r_\text{M}}_{\text{HI}}(\ _i; `)$. In SM^r_M, HI-tasks are executed up to its C^{LO}_i by its virtual deadline, similar to SM_L mode. Therefore, DBFs of HI-task in SM^r_M is computed as follows (Equation 2),

$$\forall\ _i \in\ _{\text{HI}};\ \text{dbf}^{\text{SM}^r_\text{M}}_{\text{HI}}(\ _i; `) = \left\lfloor \frac{` - D^v_i}{T_i} \right\rfloor + 1 \cdot C^{\text{LO}}_i$$

Compute $\text{dbf}^{\text{SM}_\text{H}}_{\text{HI}}(\ _i; `)$. Note that SM_H can be triggered from SM^r_M and SM^w_M. Consider the DBF computation for each case separately,

Mode switch from SM^w_M *to* SM_H: as the HI-tasks are executed up to C^{HI}_i by its actual deadline D_i similar to SM^w_M and all LO-tasks are dropped immediately in the mode-switch instant, DBFs for SM_H is simply using Equation 4 for $\forall\ _i \in\ _{\text{HI}}$,

$$\text{dbf}^{\text{SM}_\text{H}}_{\text{HI}}(\ _i; `) = \left\lfloor \frac{` - D_i}{T_i} \right\rfloor + 1 \cdot C^{\text{HI}}_i; \quad (16)$$

Mode switch from SM^r_M *to* SM_H: In this case, HI-tasks' execution demands changes from C^{LO}_i to C^{HI}_i with an increased deadline from D^v_i to D_i similar to a mode switch from SM_L to SM^w_M. So the DBF computation of SM_H is similar to SM^w_M,

$$\forall\ _i \in\ _{\text{HI}};\ \text{dbf}^{\text{SM}_\text{H}}_{\text{HI}}(\ _i; `) = \text{dbf}^{\text{SM}^w_\text{M}}_{\text{HI}}(\ _i; `)$$

So, for the feasibility test of condition D of Theorem 1, we need to take the max of the two possible demand functions.

Timing complexity of schedulability test. In Theorem 1, all four conditions need to be assessed for $\forall `$, which is unbounded number of computation. Here, we will derive upper bound for $`$ to satisfy the schedulability test for each condition. The following four lemmas upper-bound $`$ for each condition, respectively.

Lemma 1. *Let* c_1 *and* c_2 *be constants such that* $U^{\text{LO}}_{\text{LO}} \le c_1$, $U^{\text{HI}}_{\text{LO}} \le c_2$, *and* $c_1 + c_2 < \ ! ^n$. *Then the condition* A *of Theorem 1 is true* $\forall ` \ge 0$ *if it is true for* $\forall ` \le \mathcal{L}_A$ *such that,*

$$\mathcal{L}_A < \frac{c_1 \cdot \max_{i \in\ _{\text{LO}}}\{T_i - D_i\} + c_2 \cdot \max_{i \in\ _{\text{HI}}}\{T_i - D^v_i\} + R}{! ^n - c_1 - c_2};$$

where, $R = 2 \cdot\ ! ^n \cdot (\quad - \quad ^n)$

Proof. Assuming that $c_1 + c_2 < \ ! ^n$, we prove the contrapositive that if $\exists ` \ge 0$ such that the condition $A$ of Theorem 1 does not hold, then $\exists ` \le \mathcal{L}_A$ such that the condition A does not hold as well. We let $`_0$ denote such $` \ge 0$ that the condition A does not hold. That is,

$$\text{DBF}_{\text{SM}_\text{L}}(\ ; `_0) > \text{sbf}_{R\ n}(`_0); \text{ where;}$$

$$\text{DBF}_{\text{SM}_\text{L}}(\ ; `_0) = \sum_{i \in\ _{\text{LO}}} \text{dbf}^{\text{SM}_\text{L}}_{\text{LO}}(\ _i; `_0) + \sum_{i \in\ _{\text{HI}}} \text{dbf}^{\text{SM}_\text{L}}_{\text{HI}}(\ _i; `_0)$$

Now applying Equation 8, we get

$$\overline{\text{sbf}_{R\ n}}(\text{DBF}_{\text{SM}_\text{L}}(\ ; `_0)) > `_0$$

$$\overset{\text{(Eqn. 11)}}{\Longrightarrow} \overline{\text{lsbf}_{R\ n}}(\text{DBF}_{\text{SM}_\text{L}}(\ ; `_0)) > `_0$$

$$\overset{\text{(Eqn. 9)}}{\Longleftarrow} \frac{1}{! ^n} \cdot \text{DBF}_{\text{SM}_\text{L}}(\ ; `_0) + 2(\quad - \quad ^n) > `_0 \quad (17)$$

Now we simplify the computation of $\text{DBF}_{\text{SM}_\text{L}}(\ ; `_0)$,

$$\sum_{i \in \tau_{LO}} \mathrm{dbf}^{\mathrm{SM_L}}_{\mathrm{LO}}(\tau_i; \ell_0)$$
$$= \sum_{i \in \tau_{LO}} \left\lfloor \frac{\ell_0 - D_i}{T_i} \right\rfloor + 1 \cdot C_i^{\mathrm{LO}}$$
$$\leq \sum_{i \in \tau_{LO}} \frac{\ell_0 - D_i}{T_i} + 1 \cdot C_i^{\mathrm{LO}}$$
$$= \sum_{i \in \tau_{LO}} \frac{C_i^{\mathrm{LO}}}{T_i} \cdot (\ell_0 + T_i - D_i)$$
$$\leq c_1 \cdot \ell_0 + \max_{i \in \tau_{LO}} \{T_i - D_i\} \tag{18}$$

$$\sum_{i \in \tau_{HI}} \mathrm{dbf}^{\mathrm{SM_L}}_{\mathrm{HI}}(\tau_i; \ell_0)$$
$$= \sum_{i \in \tau_{HI}} \left\lfloor \frac{\ell_0 - D_i^v}{T_i} \right\rfloor + 1 \cdot C_i^{\mathrm{LO}}$$
$$\leq \sum_{i \in \tau_{HI}} \frac{\ell_0 - D_i^v}{T_i} + 1 \cdot C_i^{\mathrm{LO}}$$
$$= \sum_{i \in \tau_{HI}} \frac{C_i^{LO}}{T_i} \cdot (\ell_0 + T_i - D_i^v)$$
$$\leq c_2 \cdot \ell_0 + \max_{i \in \tau_{HI}} \{T_i - D_i^v\} \tag{19}$$

Applying Equations 17,18,19, we get —

$$\ell_0 < \frac{c_1 \cdot \max_{i \in \tau_{LO}} \{T_i - D_i\} + c_2 \cdot \max_{i \in \tau_{HI}} \{T_i - D_i^v\} + R}{1^n - c_1 - c_2}$$
$$\text{where, } R = 2 \cdot 1^n \cdot (\quad - \quad^n)$$

This implies that if the task set is not schedulable, then the demand of the task set would be greater than the supply at least one time instant before ℓ_0. So, the Lemma follows.

Lemma 2. *Let c_3 and c_4 be constants such that $\sum_{i \in \tau_{LO}} \frac{r_i \, C_i^{\mathrm{LO}}}{T_i} \leq c_3$, $U_{\mathrm{HI}}^{\mathrm{HI}} \leq c_4$, and $c_3 + c_4 < 1^n$. Then the condition B of Theorem 1 is true $\forall \ell \geq 0$ if it is true for $\forall \ell \leq \mathcal{L}_B$ such that,*

$$\mathcal{L}_B < \frac{c_3 \cdot P + c_4 \cdot Q + 2 \cdot 1^n \cdot (\quad - \quad^n)}{1^n - c_3 - c_4};$$
$$\text{where; } P = \max_{i \in \tau_{LO}} \{T_i - D_i + \frac{T_i}{r_i}\};$$
$$\text{and } Q = \max_{i \in \tau_{HI}} \{T_i - (D_i - D_i^v)\}$$

Lemma 3. *Let c_2 and c_3 be two constants such that $U_{\mathrm{LO}}^{\mathrm{HI}} \leq c_2$, $\sum_{i \in \tau_{LO}} \frac{r_i \, C_i^{\mathrm{LO}}}{T_i} \leq c_3$ and $c_2 + c_3 < 1^c$. Then the condition C of Theorem 1 is true $\forall \ell \geq 0$ if it is true for $\forall \ell \leq \mathcal{L}_C$ such that,*

$$\mathcal{L}_C < \frac{c_2 \cdot \max_{i \in \tau_{HI}} \{T_i - D_i^v\} + c_3 \cdot P + 2 \cdot 1^c \cdot (\quad - \quad^c)}{1^c - c_2 - c_3};$$
$$\text{where; } P = \max_{i \in \tau_{LO}} \{T_i - D_i + \frac{T_i}{r_i}\}$$

Lemma 4. *Let c_4 be a constant such that $U_{\mathrm{HI}}^{\mathrm{HI}} \leq c_4 < 1^c$, then condition D of Theorem 1 is true $\forall \ell \geq 0$ if it is true for $\forall \ell \leq \mathcal{L}_D$ such that,*

$$\mathcal{L}_D < \frac{c_4 \cdot \max_{i \in \tau_{HI}} \{T_i - (D_i - D_i^v)\} + 2 \cdot 1^c \cdot (\quad - \quad^c)}{1^c - c_4}$$

Note that we skip the proof of Lemma 2, 3, and 4 due to the space limitations; however, these lemmas can be proved using similar approach we presented to prove Lemma 1.

Remark 5. *We present maximum number of DBF computations, $\mathcal{L}_A; \mathcal{L}_B; \mathcal{L}_C; \mathcal{L}_D$ such that the conditions $A; B; C; D$ in Theorem 1 are satisfied for $\forall \ell \geq 0$, respectively. However, in the implementation of the test, the computation can be drastically reduced using the quick processor demand analysis (QPA) method presented in [52] with an worst case upper bound of $\mathcal{L}$.*

IV. Setting Resource Period

Resource period is an optimization problem from the fact that a larger period has the advantages of less number of budget replenishment instants and a longer continuous resource supply. In contrast, a larger period can make the scheduler fail to schedule workload due to the larger non-supply interval (*e.g.*, from Equation 4, largest '0' supply interval is $2 \cdot (\quad - \quad)$). In this section, we will derive the bounds for the resource supply period given the system workload, nominal, and critical resource budgets.

Lemma 5. *Given the workloads $\quad = \{\tau_{\mathrm{LO}}; \tau_{\mathrm{HI}}\}$ and nominal resource budget $\quad^n$, if resource period $\quad_A$ is within the following range, then the Condition A of Theorem 1 holds TRUE,*

$$\Pi^n < \quad_A \leq \frac{\quad^n \cdot (l + 2 \cdot \quad^n)}{c_1 \cdot (l + \quad) + c_2 \cdot (l + \quad) + 2 \cdot \quad^n}$$
$$\text{where, } \quad = \min_{i \in \tau_{LO}} \{T_i - D_i\};$$
$$= \min_{i \in \tau_{HI}} \{T_i - D_i^v\}; \; l = \mathrm{lcm}\{T_i \mid \tau_i \in \quad\};$$

Proof. The lemma can be proved using Condition A of Theorem 1, as follows:

$$\sum_{i \in \tau_{LO}} \mathrm{dbf}^{\mathrm{SM_L}}_{\mathrm{LO}}(\tau_i; l) + \sum_{i \in \tau_{HI}} \mathrm{dbf}^{\mathrm{SM_L}}_{\mathrm{HI}}(\tau_i; l) \leq \mathrm{sbf}_{R_n}(l)$$

By applying Fact 1 and 3, we just need:

$$\Leftarrow \sum_{i \in \tau_{LO}} \mathrm{ldbf}^{\mathrm{SM_L}}_{\mathrm{LO}}(\tau_i; l) + \sum_{i \in \tau_{HI}} \mathrm{ldbf}^{\mathrm{SM_L}}_{\mathrm{HI}}(\tau_i; l) \leq \mathrm{lsbf}_{R_n}(l)$$

Now we simplify terms in the left side of above inequality,

$$\sum_{i \in \tau_{LO}} \mathrm{ldbf}^{\mathrm{SM_L}}_{\mathrm{LO}}(\tau_i; l)$$
$$= \sum_{i \in \tau_{LO}} \frac{l - D_i}{T_i} + 1 \cdot C_i^{\mathrm{LO}} \quad \text{(Using Eqn. 3)}$$
$$= \sum_{i \in \tau_{LO}} \frac{C_i^{\mathrm{LO}}}{T_i} \cdot (l + T_i - D_i)$$
$$\geq c_1 \cdot l + \min_{i \in \tau_{LO}} \{T_i - D_i\} \tag{20}$$

$$\sum_{i \in \text{HI}} \text{ldbf}^{\text{SM}_\text{L}}_{\text{HI}}(\ _i; l)$$

$$= \sum_{i \in \text{HI}} \left\lceil \frac{l - D_i^v}{T_i} + 1 \right\rceil \cdot C_i^{\text{LO}} \quad \text{(Using Eqn. 3)}$$

$$= \sum_{i \in \text{HI}} \frac{C_i^{\text{LO}}}{T_i} \cdot (l + T_i - D_i^v)$$

$$\geq c_2 \cdot \left(l + \min_{i \in \text{HI}} \{T_i - D_i^v\} \right) \tag{21}$$

By applying Equations 20,21,6, we have:

$$c_1 \cdot (l + \ ) + c_2 \cdot (l + \ ) \leq l^n \cdot (l - 2 \cdot (\ - \ ^n));$$

$$\text{where,} \quad = \min_{i \in \text{LO}} \{T_i - D_i\}; \quad = \min_{i \in \text{HI}} \{T_i - D_i^v\}$$

$$\Leftrightarrow \quad \leq \frac{^n \cdot (l + 2 \cdot \ ^n)}{c_1 \cdot (l + \ ) + c_2 \cdot (l + \ ) + 2 \cdot \ ^n}$$

The remaining case, $^n < \ _A$, is trivial as the resource period must be greater than the resource budget to supply the budget in each period.

So, the lemma holds.

Lemma 6. *Given the workloads* $= \{\ _{\text{LO}};\ _{\text{HI}}\}$ *and nominal resource budget* ^{n}*, if the resource period* $\ _B$ *is within the following range, then the Condition* B *of Theorem 1 holds TRUE,*

$$^n < \ _B \leq \frac{^n \cdot (l + 2 \cdot \ ^n)}{c_3 \cdot (l + \ ) + c_4 \cdot (l + \ ) + 2 \cdot \ ^n}$$

$$\textit{where,} \quad = \min_{i \in \text{LO}} \left\{T_i - D_i + \frac{T_i}{r_i}\right\};$$

$$= \min_{i \in \text{HI}} \{T_i - (D_i - D_i^v)\}; \ l = \text{lcm}\{T_i |\ _i \in \ \}:$$

Lemma 7. *Given that the workloads* $= \{\ _{\text{LO}};\ _{\text{HI}}\}$*, and critical resource budget* ^{c}*, if the resource period* $\ _C$ *is within the following range, then the Condition* C *of Theorem 1 holds TRUE,*

$$^c < \ _C \leq \frac{^c \cdot (l + 2 \cdot \ ^c)}{c_2 \cdot (l + \ ) + c_3 \cdot (l + \ ) + 2 \cdot \ ^c}$$

$$\textit{where,} \quad = \min_{i \in \text{HI}} \{T_i - D_i^v\};$$

$$= \min_{i \in \text{LO}} \left\{T_i - D_i + \frac{T_i}{r_i}\right\}; \ l = \text{lcm}\{T_i |\ _i \in \ \}:$$

Lemma 8. *Given that the workloads* $_{\text{HI}}$*, and critical resource budget* ^{c}*, if the resource period* $\ _D$ *is within the following range, then the Condition* D *of Theorem 1 holds TRUE,*

$$^c < \ _D \leq \frac{^c \cdot (l + 2 \cdot \ ^c)}{c_4 \cdot (l + \ ) + 2 \cdot \ ^c}$$

$$\textit{where,} \quad = \min_{i \in \text{HI}} \{T_i - (D_i - D_i^v)\}; \ l = \text{lcm}\{T_i |\ _i \in \ _{\text{HI}}\}:$$

Note that Lemma 6, 7, and 8 can be proved using a similar approach used to prove the Lemma 5. We skipped the proofs due to page limitations.

Remark 6. *Note that, regarding* l *in the lemmas, we do not use the* $\mathcal{L}$*'s derived in the previous section as* $\mathcal{L}$ *is dependent on* . *Instead, we use a more pessimistic value:* $l = \text{lcm}\{T_i\}$*. It is safe to do so, as both the denominator and the numerator of the equations related to* *'s contain* l*.*

The MC-Budget system will be schedulable only if all four conditions of Theorem 1 are true. So, a single resource period should satisfy the periods for all four conditions bounded by Lemmas 5,6,7,8.

Theorem 2. *Given a* MC-Budget *system with workload* $=$ $\{\ _{\text{LO}};\ _{\text{HI}}\}$*, and nominal and critical resource budgets* ^{n} *and* ^{c}*. If the resource supply period* *is in the following range, then the system is schedulable:*

$$^n < \ \leq \min\{\ _A;\ _B;\ _C;\ _D\} \tag{22}$$

Proof. As the workloads are only schedulable on the system with resource budgets if four conditions of Theorem 1 hold true. Therefore, Lemmas 5,6,7,8 need to hold simultaneously implying a range of must be the smallest range among $\ _A;\ _B;\ _C;\ _D$.

V. Algorithm for Joint Determination of Deadline Tightening Parameter x for Workload and Resource Supply Period

In this section, we will jointly determine a suitable deadline shrinkage parameter x for HI-tasks and the resource supply period using the schedulability test presented in Section III and the bounds for resource period developed in Section IV.

In **Algorithm 2**, we first calculate the upper bound of feasible resource period $_{ub}$ using Theorem 2 (Line 6) for the given workload and resource budgets. Note that we choose the upper-bound of feasible resource periods instead of searching the whole range provided by Theorem 2 as the larger resource period provides better QoS minimizing budget replenishment instants. Once the resource period is calculated, we evaluate the Theorem 1 (Line 7 - 10) to check whether the workload and resource budgets are schedulable or not with current x and $_{ub}$. If all conditions in Theorem 1 are satisfied, then the algorithm return x and $_{ub}$ (Line 11) for online scheduling. In case any condition fails, we tune x using binary search following the fact that a smaller virtual deadline favors the system mode that uses C_i^{HI} of HI-tasks after mode-switch. So, if such a system mode fails to satisfy the schedulability condition for a shrinkage factor x, we reduce the factor further by updating it to $x -$ (is a shrinkage parameter for the binary search in Algorithm 2, which shrinks the search space by a factor of 0.5 in each iteration). Conversely, if a system mode with C_i^{LO} will fail, we increase the shrinkage factor by updating it to $x +$. Notice that out of the 16 possible combinations and four schedulability conditions presented in Theorem 1, only six conditions are usable for fine-tuning the deadline shrinkage factor. One combination (all TRUE conditions) returns a valid x and . Other nine combinations return failure to find a valid x to schedule the task set due to conflict in the x tuning conditions (*e.g.*, if either both C_A & C_B or C_C & C_D are false (Line 22), then there is not any

Algorithm 2: Searching x for Virtual Deadline Setting of HI-tasks and for Resource Period

Input: Workload = $\{ _1; _2; \cdots; _n\}$, LO−tasks drop-off ratio $\{r_i\}$, precision accuracy , nominal and critical resource budgets ^{n} and ^{c}.

```
1  ← 0.5 ; // step size for binary search
2  x ←  ;   // initial shrinking factor
3  while  ≥  do
4      ←  =2;
5      for each  _i ∈ |& = HI do  D_i^v ← x · D_i ;
6       _ub = min{ _A; _B; _C; _D}; // calculate
         upper bound of acceptable resource
         period for given resource budgets
         and workloads
7      C_A = Check (Condition A in Theorem 1);
8      C_B = Check (Condition B in Theorem 1);
9      C_C = Check (Condition C in Theorem 1);
10     C_D = Check (Condition D in Theorem 1);
11     if C_A ∧ C_B ∧ C_C ∧ C_D then return x, and  _ub ;
12     else if C_A ∧ C_B ∧ C_C ∧ ¬C_D then
13     x ← x −  ;
14     else if C_A ∧ C_B ∧ ¬C_C ∧ C_D then
15     x ← x +  ;
16     else if C_A ∧ ¬C_B ∧ C_C then
17     x ← x −  ;
18     else if ¬C_A ∧ C_B ∧ C_D then
19     x ← x +  ;
20     else if C_A ∧ ¬C_B ∧ ¬C_C ∧ C_D then
21     return FAILURE; // {r_i}--not acceptable
22     else return FAILURE; // x can't be found
23 end
24 return −1 ; // still possible to find a x
     with smaller
```

feasible x for such a case). After each change of x, the above steps are repeated until the algorithm converge.

Computational Complexity. In Algorithm 2, the deadline shrinkage parameter x is computed using a binary search algorithm that runs in linear time (*e.g.*, ten iterations for an accuracy precision of = 2 ^{10}). However, checking the dbf-based schedulability conditions in each iteration is pseudo-polynomial and Theorem 2 (Line 6) is linear time. Hence the overall complexity of the algorithm is *Pseudo-Polynomial*.

VI. Evaluation

In this section, we evaluate our proposed scheduling algorithm through a case study of real workloads generated by an autonomous driving system and then schedulability tests on synthetic workloads.

A. Case Study

Autonomous vehicles (AV) employ a variety of sensors to capture and understand the surrounding environment of the vehicles [21], [35]. A typical driving system is based on a modular design, which consists of sensing, perception, planning, and control [31], [27], [46]. Fig. 4 shows a generalized pipeline for modular-based autonomous driving. The captured sensor data is fed to all the perception nodes for localization, object detection, lane detection, and segmentation. Next, a sensor fusion node combines the information of the vehicle's location, surrounding objects, lanes, and open spaces together in real time. The location is also published to the global planning function to obtain a navigation route to the destination. Both the navigation route and sensor fusion results are fed to the local planning stage, which constructs a local driving space cost map and generates vehicle trajectories, and publishes it to the vehicle's drive-by-wire system. Finally, the drive-by-wire system will send control messages to ECUs through the Controller Area Network (CAN bus) to make the vehicle drive. Since the execution times of the perception tasks, *i.e.*, localization, object detection, lane detection, and semantic segmentation, vary over time and the other tasks are stable, we focus on the four perception tasks in this case study.

The autonomous driving system is implemented on a GPU workstation which consists of 28 Intel Core i9-9940X CPUs with the highest frequency at 3.3GH and 4 NVIDIA

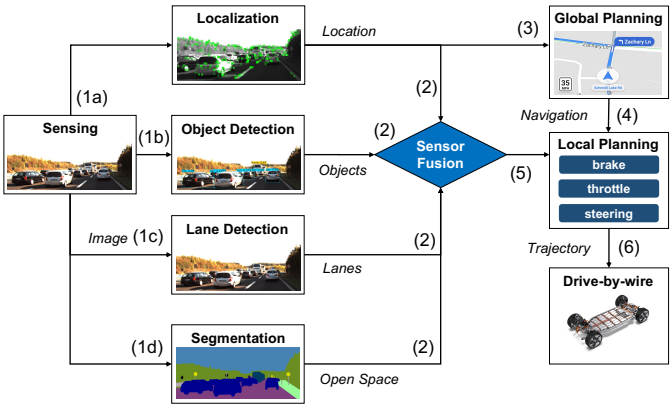


Fig. 4: A general end-to-end pipeline for modular-based autonomous driving. (Experimental setup is adopted from our previous work [34])

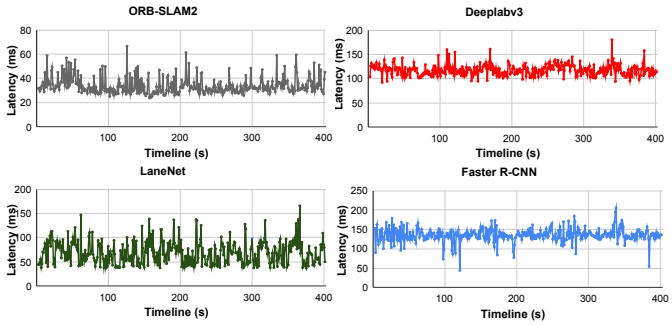
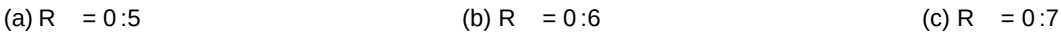
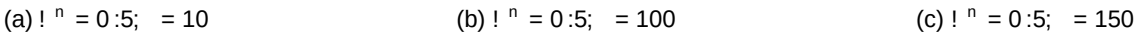


Fig. 5: The Inference latency for state-of-the-art DNN models for Localization (ORB-SLAM2), Object Detection (FAST R-CNN), Lane Detection (LaneNet), and Semantic Segmentation (Deeplabv3).



(a) R = 0:5 (b) R = 0:6 (c) R = 0:7

Fig. 6: Varying the resource budgets for different values of workload parameters while both shrinkage parameter and resource supply period obtained jointly using Algorithm 2. (= 0:5; R_C = 0:7; r_i = 0:3)



(a) ! n = 0:5; = 10 (b) ! n = 0:5; = 100 (c) ! n = 0:5; = 150

Fig. 7: Varying the workload parameters for xed values of resource budget and period. (= 0:5; R_C = 0:7; r_i = 0:3 and implicit deadlines)

GeForce RTX 2080 Ti/PCIe/SSE2 GPU cards, providing 304 TOPS in total. Each GPU card has 4352 CUDA cores, supporting 10 Giga Rays/s and 14 Gbps memory speed. The GPU-shared memory has 11GB GDDR6 with 352 memory interface widths. Additionally, the platform has 64 GB of DDR4 shared memory. Each DDR4 memory has a speed of 2666 MT/s. The libraries installed for machine learning-related applications include CUDA Driver 510.47.03 , CUDA runtime 11.6 , TensorFlow 1.15.2 , torch v1.10.1 , torchvision v0.11.2 , cuDNN 8.3.2 , OpenCV 4.2 , etc. ROS Melodic is deployed as the communication middleware. Since accuracy is essential for the autonomous driving scenario, all the DNN models are trained and tested with single precision (FP32) [1]. Four tasks are implemented for perception: ORB-SLAM2 [36] for localization; Faster R-CNN [41] for object detection, LaneNet [38] for lane detection, and Deeplabv3 [15] for semantic segmentation. The execution time for each perception task is given in Fig. 5.

Based on our case study, we have derived different workload generation parameters for abstract synthetic workload for our schedulability test in the following subsection. Speci cally, the execution time and deadline distributions are taken from the empirical results of case study.

B. Schedulability Evaluation

In this section, we rst discussed the procedure to generate synthetic workloads and resource model parameters for the simulation of the schedulability test of our proposed scheduling framework. Then we discussed the results of the tests.

Workload generation. The workload for evaluation is generated following the parameters described below.

U_W = f x=20 j 1 x 20g - Average utilization
T_i U[100; 1000] - Task periods chosen uniformly
P = 0:5 - Probability of high-critical task
R_C = 0:7 - C_i^{LO}=C_i^{HI}
R_D = f 0:7, 0:8, 0:85, 0:9g - D_i=T_i - Deadline Ratio
r_i = 0:3 - Rate of execution
R = f 0:5, 0:6, 0:7, 0:9g - c= n
n U[5; 10] - Nominal budget chosen uniformly

The number of tasks per task set is chosen to be n = 10. For each con guration, 500 task sets are generated to calculate the schedulability ratio. The task set generation procedure begins with a two empty sets of utilizations U_{LO} and U_{HI}. U_{LO} is the set of LO-mode utilizations and U_{HI} is the set of HI-mode utilizations. Next, we uniformly sample utilization u from U[0; 1] and add to U_{LO}. For each generated task, we assign its criticality to be HI with a probability P. If the task is chosen to be a HI-criticality task, then we add u=R_C to U_{HI}, otherwise we add u to U_{HI}. In this fashion, tasks are added repeatedly, to their respective sets, until average(P (U$_{LO}$); P (U$_{HI}$)) U_W 0:025. Upon addition of a task, if the average exceeds U_W +0:025, then the entire task set is discarded and the process is restarted. The task periods are sampled uniformly from [100; 1000]. Following this, the LO-mode and HI-mode WCET values are calculated. The iterative

task generation procedure used is consistent with the existing literature [6]. The resource calculation is independent of the workload. At first, a nominal budget ^{n} is chosen from a uniform distribution, and a fixed value of budget ratio is chosen from R . Following that, the critical budget ^{c} is calculated using nominal budget ^{n} and budget ratio R .

Evaluation for joint optimization of shrinkage parameter and resource period. We first experimented the acceptance test for a set of workloads where the deadline shrinkage parameter x and resource period are determined using Algorithm 2. Resulted schedulability ratio is shown in Fig. 6 for different ratio of resource budgets (R). Note that, R plays an significant rule in acceptance test. A resource model with relative small critical budget forces to fail to schedule more task sets than a 'critical budget' close to 'nominal budget'. These results imply that if $R \rightarrow 1$ (*e.g.*, signal budget supply), schedulability ratio increases significantly (which is expected). However, the necessity of dual-resource is driven from practical implication of systems as mentioned previously.

For constrained-deadline workloads, the difference between the task's period and deadline plays a vital role in the schedulability ratio and is one of the reasons to use the DBF-based test instead of the utilization-based test. As the relative deadlines are reduced with decreasing deadline ratio R_D, the performance drops in general. The results in each subplot of Fig. 6 are in line with existing results in the literature, as constrained deadlines increase the density of the workloads.

Evaluation for fixed resource model. Although we developed an optimal resource model through searching an efficient resource period for a VP for a given workload, in practice, the resource model might be fixed during and need to apply acceptance test of a workload for a fixed resource model. Such as a case is shown in Fig. 7

Resource supply period is a critical parameter and has significant effects on schedulability of workloads. In general, a lower resource period performs better due to less non-supply interval (*i.e.*, $2 \cdot (\quad - \quad)$). Each subplot in Fig. 7 demonstrates the effect of the resource supply period. A maximum difference of schedulability ratio between resource periods 10 and 200 is observed as $\sim 40\%$. Besides, resource bandwidths $!^{n}$ and $!^{c}$ play a critical role in the schedulability of workloads as the total workload utilization must be less than the resource bandwidths. Each subplot in Fig. 7 illustrates that the any taskset with $U_W > !^{n}$ is not schedulable.

VII. Related Works

Both mixed-criticality (MC) systems [47] and compositional scheduling [43] have been heavily studied over the years. An up-to-date survey on MC systems can be found in [14]. Original MC model [47] has been studied for different scheduling frameworks such as fixed-priority scheduling *e.g.*, [47], dynamic scheduling *e.g.*, [6], [20], [18]. Later MC model has relaxed for practical considerations, for instance, [13] considered running the LO-tasks to completion once it started; [7] reduced the priority of LO-tasks; [40], [44], [45] considered an elastic task model by stretching the periods and deadlines of LO-tasks; [5] analysed for the criticality of task's period; [26] analyzed imprecise MC model and [23] a budget control model; [48] considered a partition model to move LO-tasks a different processor than the critical one; [22], [24], [29] improved resource utilization while guaranteeing the execution of critical tasks. MC systems also analyzed for varying speed processors [9], [10] and for graceful degradation [8], [25], [33]. Similarly, compositional framework has been studied under different system architectures and resource model (few to mention [4], [16], [30], [37], [39], [49], [19]).

Gu *et al.* [24] presented a resource efficient MC scheduling for hierarchical scheduling framework. To provide efficient resource supply to virtual component/processor, they considered a dual-critical resource supply model called, MCPR. In MCPR, it was assumed that the supply model is always aware of the workload criticality of a VP and hence, can adjust the supply criticality according to workload criticality. Therefore, the scheduling framework for the VP is still maintain two criticality mode. In comparision with MCPR, our resource model is independent of the criticalities of workload of a VP which is more practical and used in practice [17].

Lackorzyński *et al.* [28] observed that a single supply for a VP in a hierarchical scheduling framework underperforms in the case of MC workload scheduling by the VP. To improve the performance, they have added multiple supplies for each VP and relaxed the strong isolation constraints among VP's to develop overall priority order of different critical workloads of all VPs. Leveraging the priority orders and supply model, the lower-level scheduler can provide an appropriate resource budget to each VP. However, they did not present the scheduling model and strategies for a VP, which is the main focus of this paper. Moreover, our supply model preserves the temporal isolation among VP's which is a critical concern from security perspectives [51].

Recently, Yang and Dong [50] presented a mixed-criticality model for resource budget and developed a scheduling framework for a VP. However, the paper did not consider the workload criticality model with different WCETs of a task for each criticality level. In contrast to [50], we present a generalized MC model with workload and resource budget criticality. Moreover, we consider a more general workload model, constrained-deadline workloads, and present a DBF-based schedulability test different from the utilization-based test of implicit-deadline workloads presented in [50].

VIII. Conclusion

In this paper, we have presented *MC-Budget*, the first generalized mixed-criticality system in a compositional scheduling framework that includes both resource supply criticality and workload criticality. In our model, we have considered dual workload criticalities for a constrained-deadline sporadic task model and a resource supply model with two supply budgets (*i.e.*, nominal and scarce budget) to a virtual processor, resulting in four system modes. We have developed an EDF-

VD-based dynamic scheduler and presented a DBF-based pseudo-polynomial schedulability test. We have further derived a range for resource supply periods from the schedulability of the workloads with dual-critical supply budgets. We have performed extensive simulations of the schedulability test to measure the efficacy of the proposed model on synthetic workloads and resource models. While no-such generalized framework exists in the compositional scheduling framework, our proposed MC-Budget framework lays the foundation to effectively and efficiently design modern safety-critical systems with multiple criticality levels.

Future Works. We left several research problems related to this paper for future work. *First*, in this paper, we have considered a simple sufficient heuristic to restore system mode to initial system mode by observing an idle instant in the resource replenishment instant of a virtual processor. However, it may be possible to analyze for an exact or more efficient system mode restoration instant which is open in this work. *Second*, we consider non-clairvoyance for both workload and resource supply criticality-based mode-switch instances. In the case of resource supply criticality, however, it could be possible that the OS can be clairvoyant and let the VPs know about upcoming mode-switch trigger instant. *Third*, we assumed the context switching overheads are negligible and covered by the task's WCET. However, a more practical model would be the explicit modeling of overheads in the analysis.

References

[1] Floating Point and IEEE 754 Compliance for NVIDIA GPUs. https://docs.nvidia.com/cuda/floating-point/index.html.

[2] ROS 2 Documentation. https://docs.ros.org/en/foxy/index.html.

[3] Autosar adaptive platform. https://www.autosar.org/standards/adaptive-platform/, 2022. [Online; accessed 13-October-2022].

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

[5] S. Baruah. Schedulability analysis of mixed-criticality systems with multiple frequency specifications. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10. IEEE, 2016.

[6] S. Baruah, V. Bonifaci, G. DAngelo, 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. IEEE, 2012.

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

[8] S. Baruah, A. Burns, and Z. Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *ECRTS*, pages 131–138. IEEE, 2016.

[9] S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 68–77. IEEE, 2013.

[10] S. Baruah and Z. Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. In *2014 IEEE Real-Time Systems Symposium*, pages 31–40. IEEE, 2014.

[11] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *[1990] Proceedings 11th Real-Time Systems Symposium*, pages 182–190, 1990.

[12] S. K. Baruah, V. Bonifaci, G. d'Angelo, A. Marchetti-Spaccamela, S. v. d. Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In *European symposium on algorithms*, pages 555–566. Springer, 2011.

[13] S. K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 34–43. IEEE Computer Society, 2011.

[14] A. Burns and R. I. Davis. Mixed criticality systems-a review:(february 2022). 2022.

[15] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[16] S. Chen, L. T. Phan, J. Lee, I. Lee, and O. 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.

[17] D. Dasari, M. Becker, D. Casini, and T. Blaß. End-to-end analysis of event chains under the qnx adaptive partitioning scheduler. In *RTAS*, pages 214–227. IEEE, 2022.

[18] A. Easwaran. Demand-based scheduling of mixed-criticality sporadic tasks on one processor. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 78–87. IEEE, 2013.

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

[20] P. Ekberg and W. Yi. Bounding and shaping the demand of mixed-criticality sporadic tasks. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 135–144. IEEE, 2012.

[21] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

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

[23] X. Gu and A. Easwaran. Dynamic budget management and budget reclamation for mixed-criticality systems. *Real-Time Systems*, 55(3):552–597, 2019.

[24] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin. Resource efficient isolation mechanisms in mixed-criticality scheduling. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 13–24. IEEE, 2015.

[25] Z. Guo, K. Yang, S. Vaidhun, S. Arefin, S. K. Das, and H. Xiong. Uniprocessor mixed-criticality scheduling with graceful degradation by completion rate. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 373–383. IEEE, 2018.

[26] L. Huang, I.-H. Hou, S. S. Sapatnekar, and J. Hu. 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, 2018.

[27] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[28] A. Lackorzyński, A. Warg, M. Völp, and H. Härtig. Flattening hierarchical scheduling. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 93–102, 2012.

[29] J. Lee, K.-M. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, and I. Lee. Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors. In *2014 IEEE Real-Time Systems Symposium*, pages 41–52. IEEE, 2014.

[30] H. Leontyev and J. H. Anderson. A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees. *Real-Time Systems*, 43(1):60–92, 2009.

[31] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766. ACM, 2018.

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

[33] 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 *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 35–46. IEEE, 2016.

[34] L. Liu, Z. Dong, Y. Wang, and W. Shi. Prophet: Realizing a predictable real-time perception pipeline for autonomous vehicles. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 305–317. IEEE, 2022.

[35] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[36] R. Mur-Artal et al. ORB-SLAM2: an open-source SLAM system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[37] A. Nelson, K. Goossens, and B. 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.

[38] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 286–291. IEEE, 2018.

[39] L. T. Phan, M. Xu, J. Lee, I. Lee, and O. 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.

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

[41] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[42] W. River. Arinc 653–an avionics standard for safe, partitioned systems. In *IEEE Seminar*, 2008.

[43] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, pages 2–13. IEEE, 2003.

[44] H. Su, N. Guan, and D. 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.

[45] H. Su and D. 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.

[46] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[47] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE international real-time systems symposium (RTSS 2007)*, pages 239–243. IEEE, 2007.

[48] H. Xu and A. 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, 2015.

[49] M. Xu, L. T. X. Phan, O. Sokolsky, S. Xi, C. Lu, C. Gill, and I. Lee. Cache-aware compositional analysis of real-time multicore virtualization platforms. *Real-Time Systems*, 51(6):675–723, 2015.

[50] K. Yang and Z. Dong. Mixed-criticality scheduling in compositional real-time systems with multiple budget estimates. In *2020 IEEE Real-Time Systems Symposium (RTSS)*, pages 25–37. IEEE, 2020.

[51] M.-K. Yoon, M. Liu, H. Chen, J.-E. Kim, and Z. Shao. Blinder:{Partition-Oblivious} hierarchical scheduling. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2417–2434, 2021.

[52] F. Zhang and A. Burns. Schedulability analysis for real-time systems with edf scheduling. *IEEE Transactions on Computers*, 58(9):1250–1258, 2009.