Dynamic Mapping of Mixed-Criticality Applications onto a Mixed-Criticality Runtime System with Probabilistic Guarantees

Namcheol Lee[1], Seongsoo Hong[1], and Saehwa Kim[2]
Department of Electrical and Computer Engineering, Seoul National University, Seoul, Republic of Korea[1]
Department of Information and Communication Engineering, Hankuk-University of Foreign Studies, Yongin, Republic of Korea[2]
{nclee, sshong}@redwood.snu.ac.kr and ksaehwa@hufs.ac.kr

***Abstract*—The emergence of software-defined vehicles (SDVs) introduces significant challenges in dynamically deploying services with diverse criticality semantics. To address this issue, we present a framework for the dynamic mapping of mixed-criticality applications (MCAs) onto a mixed-criticality runtime system (MCR) with probabilistic guarantees. We model an SDV service, such as a Docker container, as an MCA and provide an MCR based on a finite-state machine. We present an approach that maps the criticality levels of an MCA to those of the MCR, tracks available resources in the MCR, converts the resource demands of an MCA, and performs admission control to ensure the MCR remains schedulable. This framework enables the reliable and prioritized execution of critical SDV functions while appropriately managing less critical tasks.**

Keywords—software-defined vehicle, mixed-criticality system, deployable service, admission control, real-time scheduling

I. INTRODUCTION

As the software-defined vehicle (SDV) has emerged as an innovative technology in the modern automotive industry, it is receiving significant attention and having a substantial impact on how automotive systems are designed and manufactured. Technically defined, an SDV refers to a vehicular system in which software determines the features and functions of the vehicle. Consequently, an SDV is characterized by its reliance on software to control and manage various aspects of the vehicle's operations.

Unlike traditional vehicles, where hardware components are tightly coupled with specific functions, an SDV features a more flexible and modular architecture. This allows for easier updates, customization, and the introduction of new features through dynamic software updates.

However, such benefits come with a cost. An SDV is a highly complex distributed real-time system that must overcome numerous technical challenges, including (1) ensuring real-time performance, (2) providing fault tolerance and resilience, (3) managing complexity and scalability, (4) enabling seamless integration and interoperability, (5) managing resource allocation and optimization, and (6) dynamically deploying diverse safety-critical services with different criticality semantics.

Among these challenges, we focus particularly on the last issue since an SDV must handle various services with different levels of criticality at runtime, ensuring that the most critical functions are prioritized and executed reliably, while less critical tasks are managed appropriately.

This issue poses challenging technical difficulties. SDV services often include tasks with diverse criticalities such as mission-critical and safety-critical tasks, and their timely and reliable execution must be guaranteed at runtime via an admission control mechanism, despite the differences between the environment in which they are developed and the actual environment in which they are deployed and executed.

To address this problem, we present a sophisticated framework for admission control and dynamic mapping of mixed-criticality applications (MCA) onto a mixed-criticality runtime system (MCR) with probabilistic guarantees. In doing so, we model a deployable SDV service, such as a Docker container [1], as an MCA having domain-specific criticality semantics. We in turn provide an MCR based on a finite-state machine. Finally, we introduce an approach that maps an MCA onto the MCR during the deployment stage if admissible. To the best of our knowledge, this is the first attempt to dynamically deploy mixed-criticality vehicular services into the SDV with probabilistic guarantees.

II. PROBLEM FORMULATION

A mixed-criticality system (MCS) is a system that runs tasks with varying levels of criticality. Its objective is to guarantee the deadlines of high-criticality tasks, possibly by sacrificing low-criticality tasks [2], [3], [4]. To distinguish between high and low criticality at runtime, the MCS uses indicators called *modes*, each of which corresponds to a criticality level of the MCS. By definition, the MCS in a mode with criticality level j guarantees that all the admitted high-criticality tasks with a criticality level equal to or higher than j remain schedulable.

We apply a probabilistic model to an MCS, which transforms the MCS into our MCR. A mode in the MCR has a pre-calculated probability which is the likelihood of the MCR to be at the mode or below it. We refer to this probability as the *overall mode probability* (OMP). This enables a probabilistic guarantee of an admitted task because a task with a criticality level j is always

guaranteed as long as the MCR stays at or below mode j, according to the MCR's definition as stated above.

We define our MCR as a finite-state machine $\{Q, \Sigma, \delta, q_0\}$ with criticality levels from 0 to m, with 0 being non-critical, 1 the least critical, and m the most critical. $Q = \{q_0, q_1, \dots, q_m\}$ is the mode set where q_j is a mode with criticality level j. A mode q_j is assigned an OMP denoted by p_j. A mode is associated with the amount of available resources such as CPU bandwidth. As resources become scarce, the MCR transitions to a more critical mode. $\Sigma = \{up, down\}$ is a set of triggers for mode change: up for mode change from q_j to q_{j+1} and $down$ for mode change from q_j to q_{j-1}. δ is the transition function of $Q \times \Sigma \to Q$ and q_0 is the initial mode.

The MCR runs a set of admitted tasks $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$. A task τ_i has four attributes (l_i, c_i, t_i, d_i) that denote the criticality level, the worst-case execution time (WCET), the minimum inter-arrival time, and the deadline of τ_i, respectively [2].

An MCA, denoted by a tuple $(\mathcal{S}, \mathcal{T}')$, is a Docker container containing tasks with diverse criticality levels that range from 0 to m'. In $\mathcal{S} = \{s_0, s_1, \dots, s_{m'}\}$, s_j represents the *required probability of success* (RPS) for tasks at criticality level j with 0 being non-critical, 1 the least critical, and m' the most critical. s_0 is always 0, denoting the RPS of the non-critical level. $\mathcal{T}'$ is a set of tasks of an MCA, with each task $\tau_i' \in \mathcal{T}'$ is characterized by (l_i', c_i', t_i', d_i') just like a task in MCR.

Since the number of criticality levels of an MCA may vary depending on applications, our solution approach needs to map the criticality levels of an MCA to the modes of the MCR and perform admission control of the MCA.

III. Solution Approach

Our approach depicted in Fig. 1 maps the criticality levels of an MCA to those of the MCR, keeps track of available resources in the MCR, converts the resource demands of an MCA into the units of the resources provisioned by the MCR, and performs admission control of the MCA.

Specifically, when a deployment request of an MCA is issued, our approach invokes the mapping function $f(k)$ for each criticality level k of an MCA.

$$f(k) = \begin{cases} \min\{j \in [0, m] \mid p_j \geq s_k\}, & \text{if } p_m \geq s_k \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

Note that the $f(k)$ preserves the criticality semantics of an admitted MCA. It maps a criticality level k of an MCA to the criticality level j of an MCR that guarantees at least s_k. If the MCA has a criticality level k such that $f(k) = -1$, the MCA is rejected.

If all tasks of the MCA are successfully mapped to the MCR, our approach invokes the admission controller to perform the schedulability analysis with those tasks. If the MCR remains schedulable, the MCA is finally admitted.

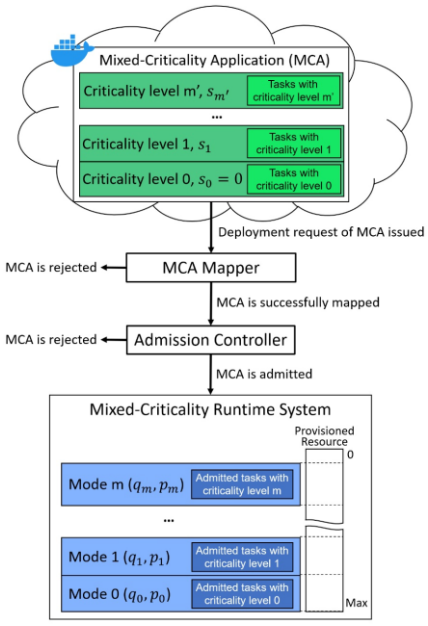

Fig. 1. Overview of dynamic deployment of an MCA onto an MCR.

The following is an example to illustrate our approach.

- MCR: $Q = \{q_0, q_1, q_2, q_3\}$ with OMPs of 0.5, 0.7, 0.8, and 0.9, respectively.
- MCA: $\mathcal{S} = \{0, 0.6, 0.85\}$ and $\mathcal{T}' = \{\tau_1' = (0, 35, 10, 10), \tau_2' = (1, 20, 5, 5), \tau_3' = (2, 10, 2, 2)\}$
- Mapping result: $f(0) = 0$, $f(1) = 1$, $f(2) = 3$
- If admitted, τ_1' will run as a best-effort task, τ_2' with a 70% guarantee, and τ_3' with a 90% guarantee.

IV. Conclusion and Future Work

In this paper, we proposed a framework for the dynamic mapping of MCAs with different criticality semantics for SDVs. This work lays the foundation for further research in the dynamic management of mixed-criticality applications in SDVs. As our research progresses, we will develop a probabilistic model for OMP, realize the admission controller, and include in our model other resources besides CPU bandwidth.

Acknowledgment

This work was supported by the Technology Innovation Program (RS-2023-00254158, "Development and Its Validation of Software-Defined Vehicle's Central Vehicle Computer Technology Applying Multiple Vehicle Domains for E/E Architecture") funded by the Ministry of Trade Industry & Energy (MOTIE, Korea).

References

[1] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014, Art. no. 2. [Online]. Available: https://dl.acm.org/citation.cfm?id=2600241

[2] A. Burns and R. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surveys*, vol. 50, no. 6, pp. 1–37, 2017.

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

[4] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Proc. 1st Workshop Mixed Criticality Syst.*, 2013, pp. 1–6.