Hybrid-virtualized architecture for time-critical applications in new generation ERTMS

Domenico Uomo, Anna Lina Ruscelli, Gabriele Cecchetti, Piero Castoldi
Scuola Superiore Sant'Anna, Pisa, Italy
domenico.uomo\annalina.ruscelli\gabriele.cecchetti\piero.castoldi@santannapisa.it

***Abstract*—The European Railway Traffic Management System (ERTMS) aims to replace the heterogeneous railway Traffic Management Systems implemented in the different countries of the European Union, promoting interoperability between national systems and, consequently, improving the efficiency of traffic management and facilitating the transit of international freight and passenger trains.**

One of the main challenges of ERTMS is meeting the high safety constraints of the railway domain, which becomes even harder considering the digitalization process of transportation systems and their integration with a wide range of services with different Quality of Service requirements. Particularly, while considering a safety-critical ICT platform, one of the most challenging issues is to guarantee time constraints of time-critical tasks in the worst case scenario. This issue can become even more challenging when trying to increase the set of functionalities the system offers, as happens in the digitalization process.

In this paper, a hybrid-virtualized architecture for ERTMS applications is proposed and preliminary tested whose aim is to operate as an ecosystem where hard real-time tasks can run together with soft real-time tasks offering an applications environment flexible, extensible, and, at the same time, suitable to meet strict temporal and high safety requirements, becoming a suitable candidate for the next generation ERTMS applications.

***Index Terms*—ERTMS, Railway, Interoperability, Virtualization, Microservices, Real-Time.**

I. Introduction

In recent years, European Rail Traffic Management Systems (ERTMS) have become one of the most important research topics in the evolution of railway systems. Within a world increasingly connected, the railway transportation has become a strategic asset for European countries since it allows an efficient and green movement of both passengers and freight. In this scope, in 2014 the European Union funded the Shift2Rail Joint Undertaking (S2R JU) and in 2021 its current evolution, Europe's Rail (EU-Rail), to boost the European railway industry and make it more competitive, also encouraging collaboration with research institutions. Their ambition is promoting the implementation of an efficient railway network spread seamlessly across Europe that raised the need to rethink the design criteria of new-generation ERTMS systems, highlighting, among other things, the use of standardized data structures and interfaces toward internal and external services [2]. A step forward in this direction has been made through the S2R project *cOmmunication Platform for TraffIc ManAgement demonstrator* (OPTIMA) [6], [11], that deployed a demonstrator for new TMS. The demonstrator consists of a system that guarantees seamless access to heterogeneous standardized data sources through standardized interfaces by internal and external services and applications, improving the TMS efficiency and interoperability.

Furthermore, the railway sector is open to technological innovations, also borrowed from other domains and the new generation ERTMS must also take into account the digitalization process that today concerns all transport systems [18], [19]. Indeed, in order to close the gap between the railway and other transportation sectors, plays an important role the exploitation of the potential offered by Information Technology (IT) systems, which allow to improve Quality of Service (QoS), efficiency, and safety of the railways applications running or interfacing toward the ERTMS, for instance integrating monitoring tasks with processes of data analytics, encompassing Artificial Intelligence (AI) and Machine Learning. In this context, the ultimate goal of ERTMS evolution is the deployment of a scalable, flexible, extensible architecture, therefore suitable to be adopted by different EU countries, and to easily support the implementation of new services in the near future. At the same time, all these characteristics must be built on top of the railway system that has strict requirements about safety, real-time responsiveness of the applications/tasks with strict temporal requirements, and predictability so as to not affect its dependability. An architecture so described impacts on the adopted implementation methods where it is possible to integrate different services with very different requirements. In this paper, leveraging the current evolution of Operating Systems (OS) with a focus on temporal requirements and the potential for flexible implementation offered by virtualization techniques, we propose a hybrid-virtualized approach for the implementation of ERTMS systems. It allows the allocation on the same machine of both soft real-time applications/tasks, with a flexible deployment leveraging the concepts of containers and microservices, and hard real-time tasks, preserving the respect of their different requirements. This proposed solution aims to enhance the adaptability of ERTMS systems, thereby promoting extensibility and flexibility in task deployment, and interoperability across different

Acknowledgment: This work has been supported by the European Rail Joint Undertaking through Project 101175856 - PhDs EU-Rail HORIZON-JU-ER-2023-01.

Europe's Rail

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Europe's Rail Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.

countries characterized by diverse operational conditions.

In particular, to provide a first testing and demonstration of the suggested approach, in this paper a preliminary deployment of a testbed about the hybrid-virtualized platform supporting different tasks/applications is described and simulated. The aim is verifying the meeting of hard real-time requirements of time-critical tasks when implemented along with heterogeneous tasks with different temporal requirements. The testbed is composed of a Linux-based OS patched with the software framework Xenomai, which introduces a different stage of execution for high-priority tasks with dedicated scheduling and policies, and Kubernetes as container orchestrator to meet the requirements of flexibility and extensibility of soft real-time applications. The experimental results show that the two kinds of tasks can coexist and collaborate to enhance the Quality of Service of the system without affecting its overall dependability.

The rest of the paper is organized as follows. Sec. II describes the current ERTMS evolution and the digitalization process of the railway sector. In Sec. III, the hybrid-virtualized architecture proposed to support the requirements of hard real-time ERTMS applications is described, while in Sec. IV the first experimental results are explained. Finally, Sec. V draws some conclusions and illustrates future works.

II. ERTMS - STATE OF ART

Currently, in Europe, different national railway TMSs are not fully interoperable due to the lack of standardized communication interfaces and data structures, hence the proposal of a common European Railway TMS (ERTMS). ERTMS aims to develop an interoperable solution that will substitute the current national implementations. This deployment will enable the creation of a seamless European railway network, increasing, therefore, its capacity and efficiency. The main component of ERTMS is the European Train Control System (ETCS), which provides operation for the Automatic Train Protection (ATP) and the Global System for Mobile Communications - Railway (GSM-R), the railway wireless telecommunication protocol [4]. The ETCS is responsible for deciding whether a train can move forward or it has to stop, providing it with a Movement Authority (MA). To compute the MA, the ETCS leverages information about the position and the speed of the trains on the track. According to the current standard, the railway track is divided into segments, called blocks, where every train occupies a block and receives the authorization to move to the next block, only if the next block is free, i.e. not occupied by another train. Three levels of implementation of ERTMS are described:

- Level 1: it allows the coexistence of traditional signaling and European signaling so to allow a smooth passage to the next generation of ERTMS, allowing the coexistence of trains with old equipment on board and trains that deploy the new implementation of ERTMS. It is based on devices distributed on the track, the *Eurobalises*, that enable the communication between trains and Radio Block Center (RBC). The devices on board of the train will receive the Movement Authority to move until the next block, and information about the track while passing on the Eurobalises. Based on that information, the trains will compute accelerations and braking curves.
- Level 2: The Eurobalises are coupled with wireless communication over GSM-R. Train and RBC will cautiously exchange information, allowing for always updated data.
- Level 3: The Eurobalises are no longer deployed, all communication is via GSM-R. Moreover, it introduces "moving blocks" for the allocation of track to trains. In this way, the blocks are dynamic and computed according to the trains' position. This allows the ETCS to provide each train with the MA until the next train on the line, rather than on the next block, optimizing the train allocation and position on the line. Currently, ERTMS Level 3 is still in the development phase.

Currently, different countries implement different levels of ERTMS. The Shift2Rail and Europe's Rail JUs arise to accelerate the process of innovation of the railway sector, promoting its competitiveness by increasing railway capacity and quality of services.

One of the most promising outcomes of the projects carried on under S2R is the OPTIMA demonstrator [10]. It deploys an Integration Layer (IL) that allows different applications, such as Rail Business Services (RBS) and Passenger Information System, to seamlessly communicate using a given data model. Moreover, the presence of an Integration Layer that decouples the different subsystems (RBS, internal and external ERTMS applications), improves the flexibility of the overall architecture, enabling the possibility of having heterogeneous applications that cooperate in a seamless and isolated way, hosted and managed in an *Applications Framework*. This platform will be illustrated in Sec. IV as the starting point of the experimental activity. As a result, it is possible to build an ERTMS that supports a wide range of functions, improving the provided QoS. Furthermore, the Integration Layer enables the introduction of different architectural approaches in the design of ERTMS systems. One of them is described in [17]. Here, an approach based on microservices is proposed, leveraging the containers for lightweight applications that can be quickly deployed, guaranteeing at the same time a high level of isolation.

A. Digitalization in railway transportation and in ERTMS

Nowadays, it is crucial to find solutions to boost the process of digitalization in order to carry the development of the railway sector and close the gap between other transportation modes, such as automotive and air transportation. Indeed, as noted in [9], the application of digital technologies can offer added value to the railway sector, which has so far failed to keep pace with other transport sectors. Furthermore, it is now essential to make rail transport a viable alternative for passenger and freight movement since, compared with the other competitors, it offers the best ratio in terms of the carried ton-kilometer per utilized energy unit, making it the most sustainable and efficient alternative [8]. The process of

digitalization of the railway sector can bring numerous benefits for passengers, freight, and infrastructures, such as:

- Real-Time information for customers, allowing them to choose the best possible itinerary for a journey.
- Optimizing and automating the planning of railway freight transportation based on knowledge of cargo location and destination, availability of resources, business orders and avoiding empty rolling stocks [16].
- Deployment of infrastructures that leverage, for instance, Artificial Intelligence, that can forecast faults based on real-time data, introducing predictive maintenance to improve QoS and Reliability-Availability-Maintainability (RAM) of both track and rolling stock [20].

Stated the crucial role of digitalization in the evolution of the railway sector, it is essential to notice that this process can happen only satisfying the condition that the dependability of the vital components remains at least as it is [15]. Thus, it becomes crucial to implement new paradigms and architectures that allow faster and more effective validation and testing to speed up the innovation process of the railway domain without neglecting its high safety standards. Under these conditions, the introduction of the most recent IT technologies in the ERTMS domain can boost the dependability of the overall railway infrastructure, also focusing on the respect of temporal requirements of time-critical applications/services/tasks. For instance, in [21] the authors deal with the problem of hazardous and rare events on railway tracks that may become numerous if we consider hundreds of thousands of kilometers of railway lines. The authors propose statistical methods and Artificial Intelligence to predict and locate those events. Data analytics and big data management thus became crucial components to make it possible, along with the coexistence and the communication between hard real-time processes, that perform monitoring, command and control, and data analysis processes, often much more complex and time-consuming but with less strict temporal boundaries.

The future implementation of ERTMS (Level 3), based on the computation of moving blocks on track, can be achieved only through an increased automation of train operations; thus, it is fundamental to provide a flexible Traffic Management System able to process data about the traffic with hard real-time constraints. An architecture with such capabilities could also enable the deployment of ERTMS Level 3 with virtual coupling, as described in [12].

III. Proof of Concept

In this paper, we propose a solution where ERTMS time-critical tasks, such as monitoring and control tasks, can be integrated with other services with less strict requirements, allowing for an architecture that is flexible and extensible but that simultaneously preserves system safety and dependability. This can be seen as the first step in deploying an ERTMS where the classical tasks of Traffic Management are integrated with other Rail Business Services, enabling strict cooperation to enhance the dependability and the quality of services of railway transportation.

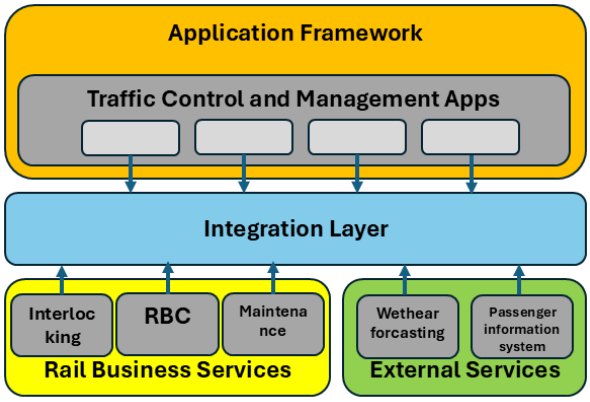

Fig. 1: OPTIMA demonstrator architecture.

The tested architecture is based on the output of the Shift2Rail OPTIMA communication platform described in [6]. In this testbed, the aim is to verify the feasibility of mapping the logical components of OPTIMA on a system able to perform time-critical tasks with negligible delay and, at the same time, guarantee support for a wide set of applications.

Particularly, if we refer to the OPTIMA architecture depicted in Fig. 1, our purpose is to create an environment suitable for the Application Framework to run, focusing on the different requirements of the applications. As proposed in the OPTIMA demonstrator, the use of virtualized components, particularly containers, is considered to deploy isolated applications. Virtualization is nowadays one of the most powerful abstractions to deploy applications, since it provides isolated environments where applications can execute, limiting the unwanted interactions among themselves, and providing, at the same time, other crucial properties like consolidation and flexibility. Particularly, containers are a form of lightweight virtual machine that share the Host Operating System (OS). In this way, every container doesn't need to carry an entire OS, unlike traditional VMs, but just the bare minimum to execute the application we need. However, we also aim to achieve a higher level of guarantees for time-critical applications, which usually container orchestrators can not provide. Most of the research about mixed-criticality systems focuses on the partitioning of resources through a hypervisor. In those cases, the hypervisor can either partition the hardware (e.g., deciding to statically allocate a processor to a process) or perform a temporal partitioning, deciding a time slot for each process to execute. The problem with this approach is due to the fact that, especially for non-critical applications, it is hard to predict how they will behave, reducing the efficiency of the services.

In this paper, we explore a different solution that does not necessarily imply the partitioning of the resources. We present an architecture where applications with soft real-time requirements are deployed through a microservices approach based on containers while hard real-time tasks are deployed as threads running directly on the host OS. With this solution, we try to build a system that can fully leverage the available resources for general-purpose tasks, and at the same time,

allows the developer to have high control over hard real-time tasks.

A. Container Virtualization

Container Virtualization is an ”Operating System-level” virtualization, meaning that, unlike traditional Virtual Machines (VM), they share the core of the OS (kernel) with the host OS and do not emulate the hardware. The result of this approach is that there's less flexibility in the choice of the guest OS but, on the other hand, containers can be allocated and managed in a more easy way keeping at the same time a higher degree of isolation with respect to the classical processes, meaning that there is limited interference among different containers. The authors in [7] demonstrate how containers have comparable performances in both computation and networking, with respect to VMs and bare-metal applications. Moreover, they demonstrate how containers need much less time to be instantiated with respect to the VMs, even in the case nested containers are created.

Nowadays, the standard de facto for container orchestration is Kubernetes. It allows to deploy and manage huge amounts of containers, spread on different physical machines, thanks to a Master-Slave architecture. Kubernetes also implements functions of monitoring for both containers and physical machines, allowing a transparent rescheduling when a failure happens, auto-scale of containers if the workload increases, and making sure that the resources are evenly distributed among containers [5].

B. Hard Real-Time tasks

For hard real-time tasks, it is necessary to take into account different considerations. Linux implements a *priority policy* that reserves 40 levels for user-space processes and 99 levels for real-time processes. It also includes a high-resolution timer and it is possible to build it with the ”preempt” option (PREEMPT_RT [1]). Despite those options, native Linux cannot provide any guarantees about scheduling and execution time. For this reason, we decided to use in our proof of concept Xenomai [13], a framework that can be integrated with Linux to provide hard real-time capabilities. Xenomai defines a high-priority execution stage. When a task is run in this stage, called *out of band (OOB)*, it cannot be delayed by any other *”in-band”* task. Thanks to this approach, Xenomai allows execution with low latency and a better Worst-Case Execution Time (WCET) compared to Linux and PREEMPT_RT as shown in [14].

Xenomai defines two principal components to be built on top of a general-purpose OS to achieve Real-Time performances (see Fig. 2):

- **Dovetail interface**: a piece of kernel code that introduces high-priority execution mode (the OOB execution stage);
- **EVL Core**: Real-time core in Fig. 2, that offers low latency services for RT applications.

Besides these components, Xenomai also introduces **LibEVL**: a set of C libraries to allow the applications to access the services of EVL Core.

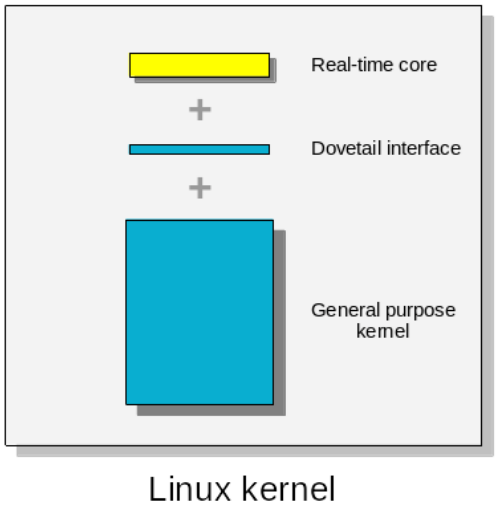

Fig. 2: Xenomai Dovetail schema [13].

C. System under Test

In the testbed we built the Xenomai Dovetail kernel together with the EVL-core libraries. Our aim is to implement a mixed-criticality system that can support containers as microservices for soft time-critical tasks, and, at the same time, guarantee the execution of hard real-time applications with no delays (or with delays less than the minimum acceptable for the considered application).

We will compare EVL threads, standard POSIX threads scheduled in the user space, and POSIX threads scheduled with Linux Real-Time priority when the system is under stress. Particularly, the time-critical threads in this scenario are periodic threads with a period of 1 second that need to read some value. This is the case of monitoring tasks that need to acquire some value, such as train speed, minimizing the delay and maximizing the predictability of the operations (i.e., reducing the variance of execution time). These threads also publish a message in the Integration Layer. Together with these time-critical tasks, we also deploy tasks with softer time requirements. Those are scheduled within containers through a container orchestrator (Kubernetes in this case). Particularly, one of the containers reads some values sent from the time-critical task on the Integration Layer. This can be seen as a task that can gather data from the monitoring applications and external applications and use those for a wide set of applications, such as predictive maintenance, improving the Passenger Information System, or enhancing train scheduling. The second container, instead, instantiates a high number of threads to simulate a stress condition of the system. Indeed, we are interested in testing the predictability of time-critical tasks when the system is stressed and there is a lack of available resources. Fig. 3 reports the schema of the testbed, where the Integration Layer is also included.

The role of IL, is to decouple tasks with different purposes and help in the flexibility of the overall architecture. In the current testbed, the IL is deployed through a simple Kafka Broker [3] that uses a publish/subscribe approach, like the IL

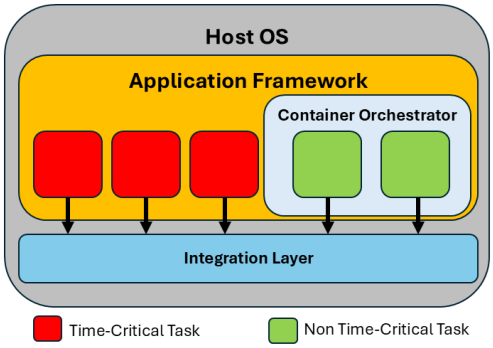

Fig. 3: Structure of the testbed.

```
** Message: 12:07:32.542: Train 0 Speed value = 302
** Message: 12:07:33.542: Train 0 Speed value = 300
** Message: 12:07:34.542: Train 0 Speed value = 304
** Message: 12:07:35.542: Train 0 Speed value = 300
** Message: 12:07:36.542: Train 0 Speed value = 302
** Message: 12:07:37.542: Train 0 Speed value = 302
```

(a) Output of the critical task (producer).

```
train speed, timestamp
304-353765840473341
average train speed:  304.0

train speed, timestamp
302-353766840474807
average train speed:  303.0

train speed, timestamp
302-353767840474605
average train speed:  302.6666666666667

train speed, timestamp
303-353768840475884
average train speed:  302.75
```

(b) Output of the general purpose task.

Fig. 4: Output of the considered applications.

from OPTIMA. In future works, the latter will substitute the current implementation with Kafka. The server we used to deploy such architecture is equipped with 8 cores Intel Xeon (2.30GHz) and 16 Gb of memory. The output of the deployed tasks is shown in Fig. 4. Particularly, in 4a it is shown the time-critical task which produces every second some value (like a monitoring task reading from the interface of a sensor). Instead, in Fig. 4b a task running in a container retrieves data from the broker and performs some computation. We deployed this very simplified scenario since the purpose of the paper is not to develop a complete application but rather to test the behavior of time-critical tasks in our environment to derive some preliminary considerations suitable for the extension of the OPTIMA demonstrator with a focus on real-time requirements.

IV. Preliminary Experimental Results

To test the deployed solution, we ran the test considering traditional POSIX threads, POSIX threads with Linux real-time priority, and finally with EVL threads. The EVL threads and real-time POSIX threads have been scheduled with the

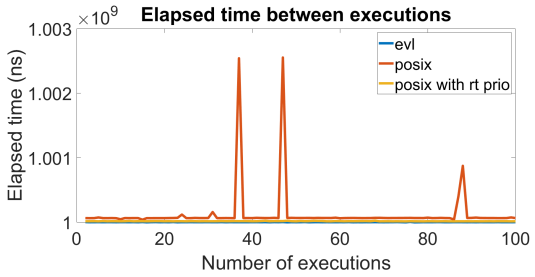

Fig. 5: Delay of the threads.

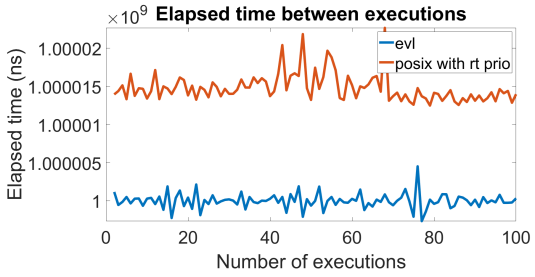

Fig. 6: Delay of the EVL and real-time POSIX threads.

same level of priority (i.e.: -1, the lowest RT priority), so the only difference between the two is that the EVL threads can access the EVL real-time services. During every execution, the threads generate 100 times a value and write it on the broker. Afterward, they print a timestamp on a file. The timestamps are then used to compute and compare the possible delay.

Fig. 5 reports the delay experienced by the three different threads when the machine hosting them is under stress. As expected, regular POSIX threads, besides having the highest delay, also have some random spikes. Thus they lack predictability. Anyway, it is interesting to notice how, scheduling the same threads with Real-Time priority, eliminates those spikes, making this implementation more suitable for critical tasks.

We can then compare EVL threads and POSIX threads with RT-prio (Fig. 6). The latter behaves in a much more predictable way if compared with the execution without priority. In any case, its latency remains higher than that of EVL threads and, although it seems negligible, it can add up and lead to considerable delays (Fig. 7).

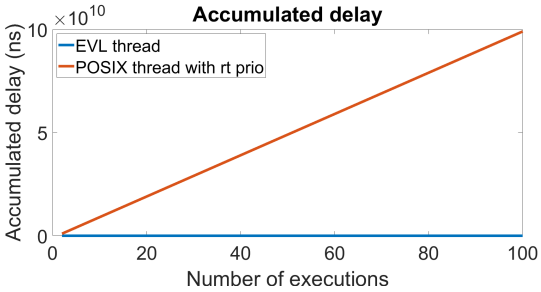

Fig. 7: Cumulative delay in the execution of EVL and RT POSIX threads.

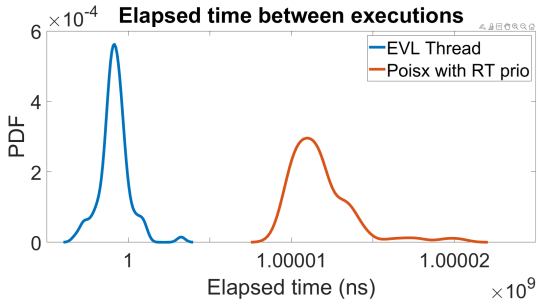

Fig. 8: Empirical PDF of the delay between two consecutive executions.

These results can impact on the predictability of the two different types of threads. In fact, we could take into account the average delay of the threads and tune the system to compensate for them. (e.g., waking up the threads some nanoseconds in advance).

In Fig. 8 an empirical Probability Density Function (PDF) of the time between two consecutive executions of the threads is derived.

From this PDF, we can see that not only the EVL thread has a lower average delay, but also the shape of the distribution is much narrower. In contrast, the distribution of the RT threads is wider and presents a long tail on the right, meaning that the execution of the thread has been delayed many times.

Finally, in Table I we compare the minimum and maximum waiting time as differences from the mean (in microseconds), and standard deviation, showing the advantages of the EVL implementation.

Thread type	Min	Max	Std
EVL thread	-2.66	+4.52	0.994
RT POSIX	-2.30	+7.88	1.81
POSIX	-88.64	+2.42	360.13

TABLE I: Comparing the time distributions of the considered tasks.

V. Conclusion and Future Works

In this paper we aim to propose a solution for the new generation ERTMS that can run simultaneously high critical tasks, typical of railway vital applications, and tasks with lower priority that can enhance the quality and quantity of services offered. To deploy such an architecture it is needed a system that is general-purpose to allow a great variety of tasks to run, to be maintained, updated and possibly extended, but at the same time, it must offers guarantees to the critical tasks. A common Linux-based OS can be the baseline to build such an architecture, but, as it is, it is not sufficient. To maximize the efficiency of the system for both general-purpose tasks and hard real-time tasks, we proposed and tested an architecture that combines a microservices approach based on containers and the Xenomai Dovetail kernel.

The tests show that this combination of frameworks can provide support for this kind of mixed-criticality environment, allowing the execution on the same system of a Kubernetes cluster together with tasks on the OS system, without affecting the predictability of the latter.

In the future we plan to integrate this architecture with the outcome of [6] project, particularly with the Integration Layer, to deliver a more complex and complete study case to demonstrate the feasibility of the railway systems digitalization along with the introduction of tools suitable to support the respect of time-critical tasks that are a crucial step towards the revitalization of railway transportation.

References

[1] https://wiki.linuxfoundation.org/realtime/start. [Accessed 14-02-2025].

[2] About S2R - Europe's Rail. https://rail-research.europa.eu/about-shift2rail/. [Accessed 25-02-2025].

[3] Apache Kafka. https://kafka.apache.org/documentation/. [Accessed 17-03-2025].

[4] European Rail Traffic Management System (ERTMS). https://www.era.europa.eu/domains/infrastructure/european-rail-traffic-management-system-ertms_en. [Accessed 18-02-2025].

[5] Scheduling, Preemption and Eviction. https://kubernetes.io/docs/concepts/scheduling-eviction/. [Accessed 25-02-2025].

[6] Shit2Rail OPTIMA project. https://www.optima-project.eu/home.aspx.

[7] M. Amaral, J. Polo, D. Carrera, I. Mohomed, Merve U., and M. Steinder. Performance evaluation of microservices architectures using containers. In *IEEE 14th NCA Int. Symp.*, pages 27–34, 2015.

[8] S. Aminzadegan, M. Shahriari, F. Mehranfar, and B. Abramovic. Factors affecting the emission of pollutants in different types of transportation: A literature review. *Energy Reports*, 8:2508–2529, 2022.

[9] B. Arnaudov. Digitalization in railway transport as a factor for improving the quality of the offered railway service. *UNWE Journal*, 2(2):145–156, 2022.

[10] G. Cecchetti, A. L. Ruscelli, C. Ulianov, P. Hyde, J. Liu, A. Magnien, M. Tavač, M. Ďuračík, L. Oneto, J. Bertolin, et al. Toward new generation railway traffic management systems: the contribution of the OPTIMA project. *Transp. Res. Proc.*, 72:3166–3173, 2023.

[11] G. Cecchetti, A.L. Ruscelli, C. Ulianov, P. Hyde, L. Oneto, and P. Márton. Communication platform concept for virtual testing of novel applications for railway traffic management systems. *Transp. Res. Proc.*, 62:832–839, 2022.

[12] C. Di Meo, M. Di Vaio, F. Flammini, R. Nardone, S. Santini, and V. Vittorini. ERTMS/ETCS virtual coupling: Proof of concept and numerical analysis. *IEEE Trans. Intell. Transp. Syst.*, 21(6):2545–2556, 2019.

[13] Philippe Gerum. Overview :: Xenomai 4. https://v4.xenomai.org/overview/. [Accessed 14-02-2025].

[14] CC Huang, Chan-Hsiang Lin, and Che-Kang Wu. Performance evaluation of Xenomai 3. In *Proc. of 17th RTLWS*, pages 21–22, 2015.

[15] A. Ozerov. Towards safer rail control, command and signalling in the context of digitization. *Dependability*, 20:54–64, 06 2020.

[16] J. Poliński and K. Ochociński. Digitization in rail transport. *Problemy Kolejnictwa - Railway Reports*, 64:137–148, 09 2020.

[17] A. L. Ruscelli and G. Cecchetti. Service orchestration and network function virtualization for information systems maintenance service in railway systems. In *Proc. of Transportation Research Arena*, 2024.

[18] A. L. Ruscelli, S. Fichera, F. Paolucci, A. Giorgetti, P. Castoldi, and F. Cugini. Introducing network softwarization in next-generation railway control systems. In *Proc. of 6th IEEE MT-ITS*, 2019.

[19] A.L. Ruscelli, G. Cecchetti, and P. Castoldi. Cloud networks for ERTMS railways systems. In *5th IEEE CloudNet*, pages 238–241, 2016.

[20] S. Sarp, M. Kuzlu, V. Jovanovic, Z. Polat, and O. Guler. Digitalization of railway transportation through ai-powered services: digital twin trains. *European Transport Research Review*, 16(1):58, 2024.

[21] I. B Shubinsky, A. M Zamyshliaev, O. B Pronevich, EN Platonov, and AN Ignatov. Application of machine learning methods for predicting hazardous failures of railway track assets. *Dependab.*, 20(2):43–53, 2020.