IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 16, 2020, accepted October 21, 2020, date of publication October 23, 2020, date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033461

Response-Time Analysis of Multipath Flows in Hierarchically-Scheduled Time-Partitioned Distributed Real-Time Systems

ANDONI AMURRIO[1], EKAIN AZKETA[2], J. JAVIER GUTIERREZ[3], MARIO ALDEA[3], AND MICHAEL GONZÁLEZ HARBOUR[3]

[1]Dependable Embedded Systems, IKERLAN Technology Research Center, Basque Research and Technology Alliance (BRTA), 20850 Arrasate, Spain
[2]Industrial Cybersecurity, IKERLAN Technology Research Center, Basque Research and Technology Alliance (BRTA), 20850 Arrasate, Spain
[3]Software Engineering and Real-Time Group, University of Cantabria, 39005 Santander, Spain

Corresponding author: Andoni Amurrio (aamurrio@ikerlan.es)

This work was supported in part by the Doctorados Industriales 2018 program from the University of Cantabria and the Spanish Government and FEDER funds (AEI/FEDER, UE) under Grant TIN2017-86520-C3-3-R (PRECON-I4).

ABSTRACT Modern industrial cyberphisical systems exhibit increasingly complex execution patterns like multipath end-to-end flows, that force the real-time community to extend the schedulability analysis methods to include these patterns. Only then it is possible to ensure that applications meet their deadlines even in the worst-case scenario. As a driving motivation, we present a real industrial application with safety requirements, that needs to be re-factored in order to leverage the features of new execution paradigms such as time partitioning. In this context we develop a new response-time analysis technique that provides the capacity of obtaining the worst-case response time of multipath flows in time-partitioned hierarchical schedulers and also in general fixed-priority (FP) real-time systems. We show that the results obtained with the new analysis reduce the pessimism of the currently used holistic analysis approach.

INDEX TERMS Schedulability analysis, time partitioning, hierarchical scheduling, distributed systems, safety, industrial application.

I. INTRODUCTION

The increasing demand of computing capacity as well as the requirement to meet safety and application-specific standards that mandate adherence to complex architectural patterns, have recently lead to cyberphysical systems becoming more and more complex [1]. These systems are typically distributed real-time systems in which it is essential to guarantee that computation is performed within a bounded time. To achieve this, deadlines imposed on the software must not be exceeded even in the worst-case situation. One way to ensure that these deadlines are always met is to accurately calculate worst-case response times, or upper bounds on them [2]. Schedulability analysis techniques include mathematical methods that allow obtaining such response times considering the worst-case scenario.

Recently, the design of safety critical systems has evolved towards partitioned architectures that guarantee the isolation among components, so that any timing error in a non-critical part does not jeopardize critical components of the system [3], [4]. This isolation can be obtained by partitioning [5], for which hierarchical schedulers play a main role: a guaranteed availability of execution time is achieved thanks to the primary scheduler, and other scheduling policies implemented in the secondary scheduler can provide the required flexibility. Moreover, the need of implementing redundant architectures for safety certification [6] has provoked that logical architectures exhibit complex multipath execution patterns.

A. RELATED WORK

The real-time research community has produced a vast number of schedulability analysis techniques, as observed in [7]. Regarding hierarchical schedulers, different analysis techniques can be used in order to obtain their worst-case response times: for instance the holistic approach, proposed by [8] provides upper bounds of the worst-case response time in hard real-time systems, and in [9] this technique was extended to multipath flows. In [10] the holistic approach is improved for Fork-Join distributed real-time tasks and a particular network (FTT-SE) by using code paralelization techniques.

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

However, the holistic approach is very pessimistic because it considers that all tasks are independent, as we will show later on.

The offset-based analysis [11] reduces the pessimism of the holistic approach by considering the relationships among tasks in the same flow through the use of offsets, which represent lower bounds on the task release times. Despite the improvements of this approach, it has only been proposed for linear flows in hierarchically scheduled time partitioned systems [12], which limits its applicability to complex multipath system models like the industrial use-case addressed in this paper. Another interesting contribution on the schedulability analysis of time-partitioned systems was proposed in [13], where a response time analysis method is proposed for mixed criticallity applications. This technique is not applicable to our work since it does not allow tasks of the same flow to be located in different partitions, which is a common feature in partitioned systems that make use of an Input/Output (I/O) partition to handle communications with other nodes of a distributed architecture.

In [14] authors present a method to schedule parallel real-time tasks in mixed-criticality systems. This scheduling algorithm is based on task-to-core allocation and the assignment of virtual deadlines to obtain a schedulable solution (if one exists) for an input task set. However, the system model they target is more restrictive than ours, considering that they assume that synchronization overheads are negligible, while in our work network latencies can be included in the analysis. They do not include time partitioning nor hierarchical scheduling in their system model, which are key features in our motivating industrial use-case.

A special mention should be done on Assertion Based Verification techniques, which include formal verification methods where designs are verified against certain assertions. In [15] a unified framework for executing static and dynamic verification [16] for embedded systems design is presented. One common static verification method is Timed Automata Models [17] which allow determining whether or not temporal requirements are met. A popular tool that is used in this context is UPPAAL [18], which can be used to assert whether a system will meet its timing constraints. A recent research work [19] shows how ARINC-compliant time-partitioned schedulers can be modeled following this method. However, the response time analysis techniques that we explore in this paper have several advantages that may be important in some classes of systems. On the one hand, the result of the analysis, i.e., worst-case response times, are very intuitive numbers for engineers who want to assess how far or close the response times are from the deadlines. On the other hand, response time analysis can be used to easily analyze complex systems, which may have rather large timed automata models. Besides, for complex models the response time analysis is considerably faster tan applying model checking techniques. Readers are encouraged to read [20], where authors perform a deep study on how different abstractions affect the performance analysis of real-time systems, concluding that there is not an abstraction that always outperforms the others. These reasons motivate the extension of response time analysis in hierarchical time-partitioned systems to multipath flows.

B. CONTRIBUTION

In this paper, we extend the offset-based analysis for partitioned systems to two particular event handlers in the multipath flow model: (1) Fork, in which a task or a workload event simultaneously activates several tasks, and (2) Join, in which a task is executed only when all the triggering events have arrived. We will show how the selected industrial use case, as well as applications with similar features, can be modeled and analyzed by using the proposed technique. Furthermore, this analysis technique is not constrained to being applied only to partitioned systems, but it is really an extension of the offset-based analysis, and therefore it can also be used in the analysis of general fixed-priority distributed systems. We will highlight how the pessimism of the holistic analysis is mitigated by our new analysis. This analysis shall provide the worst-case response times for the railway signaling application we are currently working with, in order to help the engineering team in early development stages to get a notion on how their designs are satisfying temporal constraints imposed by certification authorities [21]. Of course, the analysis will provide the required guarantees in the final development stages. For modeling the industrial use-case, we have selected MAST (Modeling and Analysis Suite for Real Time Applications) [22], which implements a model aligned with the OMG MARTE standard [23], as well as a set of tools that will support our work.

The rest of this paper is organized as follows. Section II describes the signaling application under analysis and the motivation of the work. Then, Section III presents the system model, which is used to model the application in Section IV. Section V addresses the proposed response time analysis, which is validated by applying it to the industrial use case in Section VI and to general distributed systems in Section VII. Finally, Section VIII draws the conclusions and future work.

II. INDUSTRIAL USE CASE

The use case addressed in this paper is based on a European Rail Traffic Management System (ERTMS) signalling application implemented by a railway manufacturer. ERTMS [24] is a standard resulting from an important industrial European project which aims to create a common system for traffic signaling and management in railways. This standard has two main components: GSM-R (Global System for Mobile Communications - Railway), which is in charge of wireless communications, and ETCS (European Train Control System), which performs signaling and supervision duties for traffic management. On-board ETCS is the distributed and safety-critical equipment within the vehicles that performs computation tasks. It is connected to other on-board subsystems (such as switches, sensors or other processing units performing secondary functionalities) by point-to-point connections through interfaces specified in the standard.

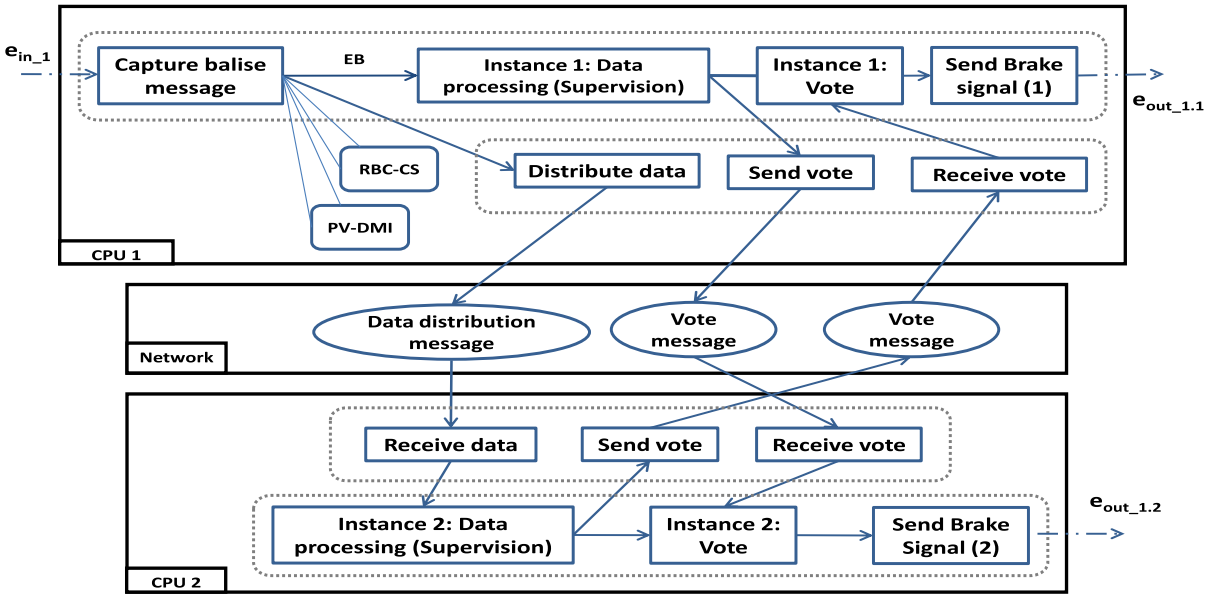


FIGURE 1. Arquitecture of EB functionality (RBC-CS & PV-DMI not depicted for the sake of clarity).

Trains receive driving indications and restrictions from the railway infrastructure, such as balises, and other interfaces, for instance radio connections with centralized control centers (Radio Block Centre, RBC). The main duty of the on-board ETCS subsystem is to provide drivers with all the information needed for a safe driving, as well as supervising that the train is traveling respecting the received instructions at all times.

A. RAILWAY SIGNALLING APPLICATION OVERVIEW

In order to perform driving supervision, different safety functionalities must be carried out within a bounded time. In this work, the considered signalling application performs three functionalities: (1)Applying the Emergency-Brake (EB functionality), (2) RBC communication session establishment (RBC-CS functionality) and (3) Parameter visualization in the Driver-Machine Interface (PV-DMI functionality). Each of these functionalities, as well as related software and hardware, need to be certified for a certain integrity level meeting IEC 61508 safety standard's requirements.

According to this standard, safety functionalities must reach SIL4, while hardware equipment reaches just SIL2 because of the trade-off among the execution platform supplier and the railway manufacturer, concerning mostly economic reasons. That is why another mechanism needs to be implemented in order obtain SIL4 compliant functionalities: in this case a Dual Modular Redundancy architecture is used, where results obtained at each instance are voted following a 1oo2 scheme [25].

Figure 1 describes in detail the functional architecture of the EB functionality for a SIL4 implementation. The other two functionalities (RBC-CS and PV-DMI) follow the same architecture, although they are not completely shown due to lack of space. A workload event, which is triggered when the train runs through a balise, activates a sequence of functions that are briefly explained as follows: (1) the message coming from the balise interface is captured, (2) supervision is then performed at the first instance in CPU-1 by processing the captured data, and (3) concurrently, data are distributed to the second instance executing in CPU-2 for redundant processing, (4) results obtained from both supervision functions are interchanged, so that they are independently voted on each processor, and (5) the voting results are sent twice to an external subsystem, which will command the needed actions. Deadlines are imposed on the brake activation events (e_{out}) referred to the signal reception (e_{in1}). Missing these timing requirements would lead to a system failure and a train crash might happen.

B. MOTIVATION

To this day, train manufacturers have indeed implemented the application described above. However, its implementation is based on a simple cyclic executive where all functions composing functionalities are called periodically, even if they do not have useful work to do. This results in an inefficient resource usage which has become a major concern for the train company; 100% of processor is dedicated to the execution of the application. The current implementation does not support adding extra functionalities easily nor the possibility of integrating it with other applications while guaranteeing the system's integrity. Worst-case response times are estimated now by testing the sequential application code to ensure that deadlines are met, and integrating this application with others would require more sophisticated response time analysis techniques than just testing.

Due to these reasons, manufacturers seek a complete system re-design, making use of modern techniques such as those used in avionics or the automotive industry mentioned in the previous section. ARINC-like software architectures, which are widely used in critical and mixed-criticality

systems' design [26]–[28], provide the capacity to integrate several safety and non-safety applications through strict partitioning, also for distributed architectures (which is in fact our case, being a redundant architecture). In these systems processors make use of a hierarchical scheduler; a primary cyclic scheduler provides temporal isolation among different components, and within each of these partitions, threads are processed following a preemptive fixed-priority scheduling. A first attempt towards the re-factoring of critical control applications in the railway domain is presented in [4], which proposes the implementation of novel architectures that ensure the execution of safety critical components together with non critical ones.

III. REAL-TIME SYSTEM MODEL

This work considers a distributed architecture composed of processors connected through one or more communications networks. Processors can be heterogeneous in terms of computation speed and memory resources. They provide hardware and software resources for task execution: sensors, actuators, memory, programs, libraries…They also host a real-time operating system that, among many other features, provides the capability of time-partitioning, where a fixed priority policy is used to schedule tasks within each partition. Regarding communications, we assume real-time networks in which worst-case message latencies can be measured or estimated. Normally, response times can be obtained in networks by using the same techniques used for processors. However, in the context of this work and with no loss of generality, we will assume networks as black boxes where each message is characterized by a minimum and a maximum latency. With this, we guarantee the applicability of this model and its associated analysis tool to any distributed system whose network latencies are, one way or another, available. The compositional approach [29] allows us to integrate the network-level analysis with processor-level analysis.

The logical architecture is composed of distributed *end-to-end (e2e) flows*, which can be activated by either periodic or sporadic events with a minimum interarrival time. These e2e flows contain a kind of event handler called *step*, which represents an operation being executed by a schedulable resource (a task or message) in a processing resource (a processor or network) with certain scheduling parameters. Steps are activated from an input event and generate an output event when they finish their execution. In the context of this work, two other event handlers are considered to represent the multipath case: *Fork* and *Join*. The *Fork* event handler generates an event in each output whenever an input event is received. Similarly, the *Join* event handler generates an output event when all the associated input events have arrived. Therefore, the Γ_i e2e flow is composed of m steps. Each instance of this flow is activated by a workload event (periodic or sporadic) arriving with a minimum separation of T_i. Each event handler except the first is released when its predecessor handler has finished its execution or, for the *Join* event handler, when all its predecessors have finished. The j-th step

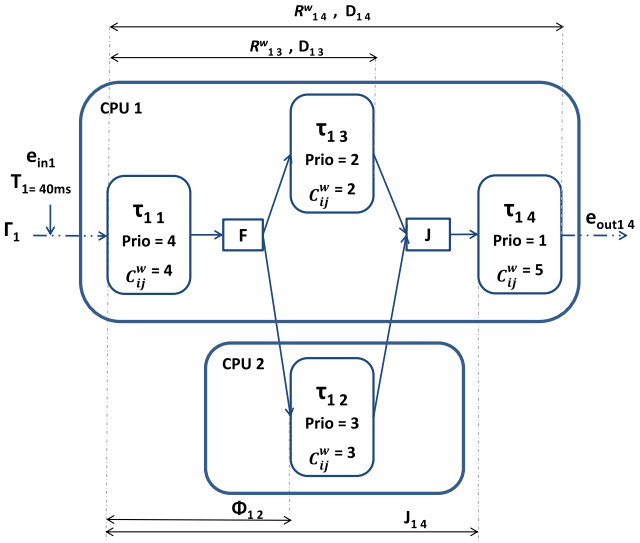


FIGURE 2. Distributed multipath e2e flow.

within the flow Γ_i is denoted as τ_{ij}, and it has a worst-case execution time C_{ij} and a best-case execution time C^b_{ij}. *Fork* and *Join* event handlers do not have runtime effects.

Within an e2e flow, each step may have a global deadline D_{ij} relative to the nominal activation time of the workload event (t_{in}, for the n-th instance). We call end-to-end deadlines those timing requirements set on the final steps of e2e flows. For each instance of a step τ_{ij}, the difference between its completion time and the nominal activation time of the workload event that triggered that instance of its e2e flow is called response time, and it is obtained by schedulability analysis techniques. The worst-case response time is denoted as R^w_{ij}, and similarly the best-case response time is denoted as R^b_{ij}.

Workload events activating e2e flows and internal events activating handlers may have a release jitter. Therefore, any step τ_{ij} within an e2e flow may suffer release jitter up to a maximum of J_{ij}. Steps may also have an initial offset ϕ_{ij}, which is the minimum release time of the step τ_{ij}, relative to the nominal activation instant t_{in}. Therefore, the release time for that step will be in the range of $[t_{in}+\phi_{ij}, t_{in}+max(\phi_{ij}, J_{ij})]$.

For periodic e2e flows, deadlines, jitters and offsets can be larger than the periods of the e2e flows.

Figure 2 shows a simple system model describing the elements of a multipath e2e flow. A workload event represented by a vertical down-pointing arrow (e_i) forks and activates two steps. Their output events then join to activate a final step. Horizontal arrows represent precedence relations among steps or control flow event handlers.

In this work, processors make use of a real-time hierarchical scheduler, where a timetable driven scheduling policy (time partitioning) is used for the primary scheduler, and pre-emptive FP for the secondary scheduler. A temporal partition i is a set of n partition windows Win_{ik} within a *Major Frame* (MAF) that is cyclically repeated. Each partition window is defined by the start time S_{ik} relative to the MAF and its duration L_{ik}. Hence, partition windows contained in a

TABLE 1. Summary of notation.

Name	Notation	Description
Processor	$CPUi$	CPU i, providing HW/SW resources for task execution
Major Frame	MAF	Period of the cycle used by the primary scheduler, containing the temporal partition windows
Partition	P_i	Temporal partition i, composed by a set of partition windows
Partition windows	win_{ij}	Time window j in partition i
Start Time	S_{ij}	Start time of the window number j in partition i, within the MAF
Duration Time	L_{ij}	Time lenght of the window j in partition i
e2e flow	Γ_i	End-to-end flow i, which represents the set of activities and actions executed in the system in response to a workload event or an event arrival pattern
Workload event	e_i	Event that triggers the e2e flow number i
Period	T_i	Activation Period or minimum interarrival time of the event that triggers the e2e flow i
Event Handlers	-	Represent actions that are activated by the arrival of one or more events, and that in turn generate one or more events at their output
Fork	F	Generates one event in each of its outputs each time an input event arrives. Does not have runtime effects.
Join	J	Generates an output event when all of its input events have arrived. Does not have runtime effects.
Step	τ_{ij}	Step j in e2e flow i. It represents the execution of an operation in a processing resource, with some given scheduling parameters
Deadline	D_{ij}	Deadline of step number j in e2e flow number i. It is a timing requirement that must be satisfied, representing the maximum response time of the step, relative to the nominal activation time of the workload event
Priority	$Prio_{ij}$	Priority of step j in e2e flow i. It is a scheduling parameter valid in the context of each partition/processor. Highest value means highest priority.
Offset	Φ_{ij}	Offset of step j in e2e flow i. It is the minimum activation time of a step, relative to the nominal activation time of the workload event
Jitter	J_{ij}	Maximum jitter that may be experienced in the activation of step j of e2e flow i. It is the maximum time that may be added to the activation time of each step instance.
Worst-case execution time	C_{ij}^w	Worst-case execution time of step j in e2e flow i
Best-case execution time	C_{ij}^b	Best-case execution time of step j in e2e flow i
Worst-case Response time	R_{ij}^w	Worst-case response time of step j in e2e flow i. It is the maximum time elapsed from the nominal activation of the workload event and the completion of the step.
Best-case Response time	R_{ij}^b	Best-case response time of step j in e2e flow i. It is the minimum time elapsed from the nominal activation of the workload event and the completion of the step.

temporal partition are defined as follows: $Win_{ik} = \{S_{ik}, L_{ik}\}$, being k in the range 1..n.

Table 1 contains a summary of the notation employed to describe the real-time system model that describes the temporal behavior of an application. Thus, it is the notation used to model the railway signaling use-case addressed in this paper.

The system model considered in this work is compliant with MAST [22], which is a model and also a set of tools that allow describing the temporal behaviour of real-time systems, and includes several algorithms for schedulability analysis, simulation and scheduling parameters assignment. MAST implementation is open source under GPL license and it has been successfully integrated in other model-based development tools as part of a MDE (Model Driven Engineering) methodology [30]. The second version of the metamodel, MAST 2 [31], adds new modelling elements and scheduling policies, such as ARINC 653 compliant time partitioning. The terminology used in the following lines is aligned with the OMG's MARTE standard.

IV. MODELLING THE INDUSTRIAL USE CASE

Considering the real-time model defined in Section III, the industrial use case described in Section II can be modeled as shown in Figure 3. As can be seen, the complete subsystem is modeled as a single multipath e2e flow triggered by the workload event e_i representing the reception of a signal from the balise, which can in turn activate several functionalities. The signal-capturing task is represented by the step $\tau_{1\,1}$. Similar to Figure 1 and for simplicity, only the real-time model of the EB functionality has been depicted in detail, even though the three functionalities presented in previous sections are going to be analyzed. In each processor two kinds of functions can be distinguished in response to the workload event: (1) those in charge of processing the information, voting and commanding the brakes for the EB functionality ($\tau_{1\,2}$, $\tau_{1\,12}$, $\tau_{1\,13}$, $\tau_{1\,6}$, $\tau_{1\,10}$, $\tau_{1\,11}$), and (2) the ones dealing with the communications to send and receive the messages ($\tau_{1\,3}$, $\tau_{1\,5}$, $\tau_{1\,9}$, $\tau_{1\,4}$, $\tau_{1\,7}$, $\tau_{1\,8}$). According to this classification, the steps will be allocated in two separate partitions for each processor: P1 hosting the steps for application processing and the I/O partition, P2, where the communication driver is located hosting the communication steps. For the EB functionality, three messages are sent through the network, represented by steps $\tau_{1\,38}$, $\tau_{1\,39}$ and $\tau_{1\,40}$. The other two functionalities follow the same scheme as the EB one: steps $\tau_{1\,14}$ to $\tau_{1\,25}$ model the RBC-CS functionality and steps $\tau_{1\,26}$ to $\tau_{1\,37}$ model the PV-DMI functionality. Steps $\tau_{1\,41}$ to $\tau_{1\,43}$ and $\tau_{1\,44}$ to $\tau_{1\,46}$ represent the messages transmitted in both functionalities.

As can be seen, the proposed task and partition allocation is fixed, while other configuration parameters such as partition

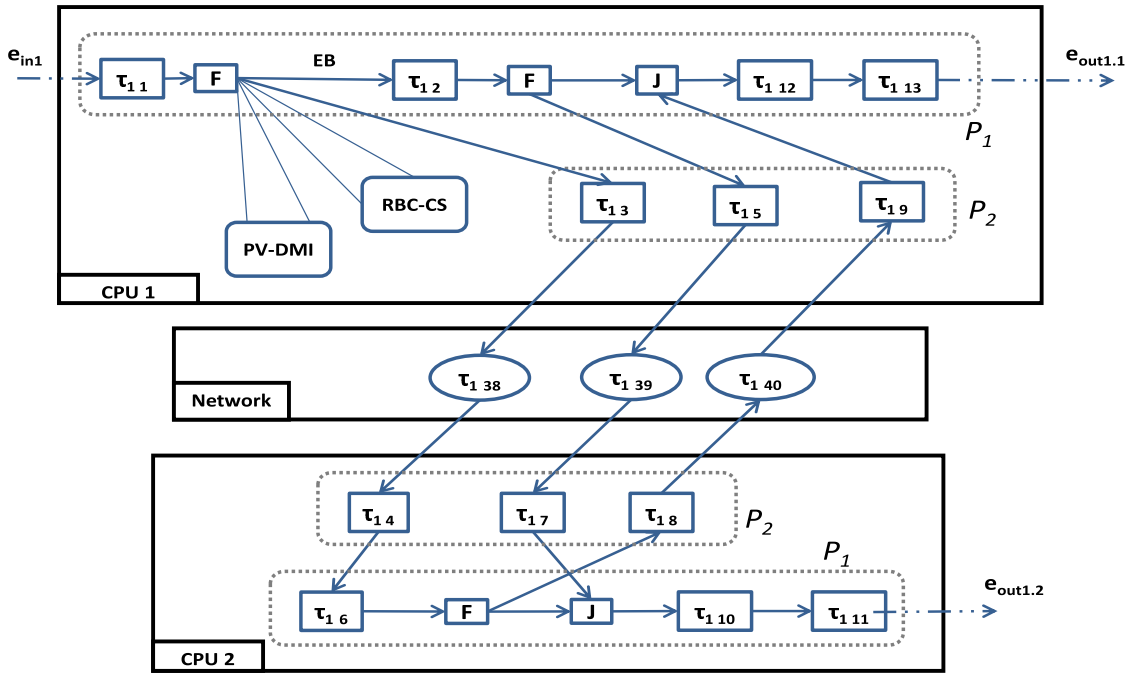


FIGURE 3. Real-time application model (RBC-CS & PV-DMI not depicted for the sake of clarity).

sizes, task priorities or execution times have values that will be given in Section VI, devoted to the use case evaluation. However, it is important to note that no matter the values assumed, the schedulability of the whole system is going to be analyzable through the developed tool if a system following a model like the one shown here is implemented.

V. RESPONSE TIME ANALYSIS

The multipath event flow architecture can be analyzed using the technique proposed in [9], based on the holistic analysis [8], which provides upper bounds on the response times, but it is known to be very pessimistic. On the other hand the offset-based analysis for time-partitioned systems [12], which provides tighter upper bounds on the response times, is only suitable for linear e2e flows, i.e., with no *Fork* or *Join* event handlers.

In this section, we will present the equations used to propagate offset and jitter values for the offset-based response time analysis of linear e2e flows. Then, we will extend these equations to the multipath model, with the objective of taking advantage of this less pessimistic analysis technique.

A. OFFSET-BASED ANALYSIS IN LINEAR e2e FLOWS

The offset-based analysis for time-partitioned systems [12] is based on the analysis for heterogeneous systems [29], where the response time analysis is iteratively applied to each step τ_{ij} independently, using as input information an inherited offset ϕ'_{ij} and an inherited release jitter J'_{ij}. These inherited offsets and jitters are calculated according to equations (1) and (2) below, and they depend on the initial offset, ϕ_{ij}, and jitter, J_{ij}, and also on the worst- and best-case response times of the predecessor step in the e2e flow (τ_{ij-1}). As a result, the worst-case response times (R_{ij}) are obtained. A lower bound estimation of the best-case response time R^b_{ij} can be calculated by adding the best-case execution times of each step in the e2e flow up to τ_{ij}.

$$\phi'_{ij} = max(R^b_{ij-1}, \phi_{ij}) \tag{1}$$

$$J'_{ij} = J_{ij} + max(R^w_{ij-1}, \phi_{ij}) - \phi'_{ij} \tag{2}$$

B. ANALYSIS OF THE MULTIPATH MODEL

In the multipath model considered in this paper, we can have *Join* and/or *Fork* event handlers. These flow control handlers have no runtime effects (i.e., their execution time is zero) as they are just structural artifacts for handling events in MAST [22]. Therefore, in the case of a *Fork* event handler, the inherited offsets and jitters obtained with equations (1) and (2) at the input are directly propagated to all of its outputs. For this case, the analysis tools only need to be updated to perform this propagation to multiple outputs.

However, for the join event handlers, the previous equations are no longer valid as there is more than one predecessor step. Now, we will show how to correctly propagate the inherited offsets and jitters values.

Lemma1: For the analysis of a join event handler in a multipath e2e flow, the inherited offset for step τ_{ij} that is the successor of the event handler is:

$$\phi'_{ij} = max(max_{\forall \tau_{ik} \in pred(\tau_{ij})} R^b_{ik}, \phi_{ij}) \tag{3}$$

where $pred(\tau_{ij})$ is the set of predecessor steps of the *Join* event handler.

Proof. The inherited offset of τ_{ij}, ϕ'_{ij}, is its minimum start time. By definition of the task model, this step cannot start before its initial offset, ϕ_{ij}, has elapsed since the arrival of the

workload event. In addition, it cannot start before the *Join* event handler has generated its output. For this output to be generated it is necessary that all the *Join* input events have been generated. The minimum generation time for each of them is the best-case response time of their predecessor step. Therefore, by taking the maximum of all the R^b_{ik} for all k in the predecessors of the *Join* handler we obtain the best-case generation time of its output event. Therefore, the lemma follows. □

Lemma 2: For the analysis of a *Join* event handler in a multipath e2e flow, the inherited jitter for step τ_{ij} that is the successor of the event handler is:

$$J'_{ij} = J_{ij} + max(max_{\forall \tau_{ik} \in pred(\tau_{ij})} R^w_{ik}, \phi_{ij}) - \phi'_{ij} \qquad (4)$$

where $pred(\tau_{ij})$ is the set of predecessor steps of the *Join* event handler.

Proof. The inherited jitter of τ_{ij}, J'_{ij}, is the difference between its maximum and minimum start time. The minimum start time is given by the inherited offset, ϕ'_{ij}. By definition of the task model, τ_{ij} cannot start before its initial offset, ϕ_{ij}, has elapsed since the arrival of the workload event. In addition, it cannot start before the *Join* event handler has generated its output. For this output to be generated it is necessary that all the *Join* input events have been generated. The maximum generation time for each of them is the worst-case response time of their predecessor step. Therefore, by taking the maximum of all the R_{ik} for all k in the predecessors of the *Join* handler we obtain the worst-case generation time of its output event. According to the task model, τ_{ij} may suffer an additional jitter bounded by its initial jitter, J_{ij}. In the worst case, there will be a delay equal to J_{ij} applied to the worst generation time of the *Join* output event. Therefore, the lemma follows. □

Equations (3) and (4) are generalized expressions for the propagation of inherited offsets and jitters, which are the basis for applying offset-based analysis. Thus, existing analysis techniques can be directly applied to the model just by updating the propagation of offsets and jitters according to these equations. Furthermore, this solution can be applied not only to hierarchical time partitioned systems but also to systems scheduled only by FP policy, since this would be a particular case where there is only one partition taking up the whole CPU.

C. SIMPLE EXAMPLE

In order to explain how response-time analysis can be applied to our industrial use case, a simple example is proposed. This allows analyzing the key features, say fork-join multipath flows, considering a reduced number of steps and removing the communications network without loss of generality. A simple example that can be solved manually provides the necessary intuition behind the theory.

Figure 4 shows a simple e2e flow consisting of four steps, two *Fork* and two *Join* event handlers. This e2e flow is distributed in two processors and executed in a single partition per processor. Both partitions have a 40ms MAF and

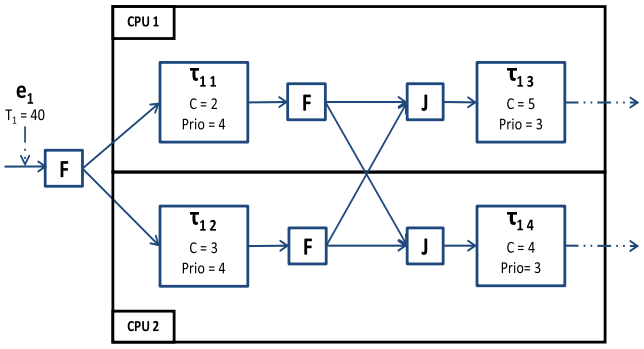


FIGURE 4. Simple example.

TABLE 2. Analysis results of the simple example (times in ms).

Step	Initial				Iteration 1			
	ϕ'	J'	R	R^b	ϕ'	J'	R	R^b
τ_{11}	0	0	2	2	0	0	**12**	**2**
τ_{12}	0	0	3	3	0	0	**13**	**3**
τ_{13}	3	0	8	8	3	10	**28**	**8**
τ_{14}	3	0	7	7	3	10	**27**	**7**

are composed of two partition windows: $Win_{11} = \{0, 10\}$ and $Win_{12} = \{20, 10\}$. The worst-case execution times in milliseconds and the priorities of the steps (the higher the number means a higher priority) are also given in Figure 4. For simplicity, we assume that best-case and worst-case execution times are the same.

Table 2 shows the results of the analysis of the simple example using the time-partitioned analysis technique with the propagation of offsets and jitters updated according to Equations (3) and (4). The initial values for the analysis are shown, using the best-case results as the initial values for the worst-case response times. In addition, since the analysis technique is iterative, we show the results for the first iteration. A second iteration gives the same results and thus no more iterations are required. Notice the effect of time-partitioning in the response times: due to those 10ms windows in which the execution is interrupted in every cycle, the analysis must consider the worst-case scenario where the activation of all steps must be deferred until this unavailable window finishes and execution can be resumed. For more details on the effects of time partitioning refer to [12].

D. IMPLEMENTATION AND TOOLS

The analysis technique for time-partitioned systems [12] has been implemented in a prototype tool that includes the slanted offset-based technique [32]. We have extended this tool according to the results of this section to integrate the analysis of multipath e2e flows (with *Fork* and *Join* handlers), including the effects of the network as a black box. In addition, we have implemented the analysis in the open-source tool MAST (https://mast.unican.es/).

MAST version 1.5.1.0 supports multipath e2e flows with *Fork*, *Join* and also *Merge* control flow event handlers. *Merge* is an event handler that generates its output event every time that any of its input events arrives. The analysis of the *Merge* event handler is directly supported by the proposed equations, since this event handler can be transformed into a linear

TABLE 3. Response time analysis of a train signalling application (times in μs).

Functionality	**Emergency-brake application - EB**												
τ_{ij}	$\tau_{1\,1}$	$\tau_{1\,2}$	$\tau_{1\,3}$	$\tau_{1\,4}$	$\tau_{1\,5}$	$\tau_{1\,6}$	$\tau_{1\,7}$	$\tau_{1\,8}$	$\tau_{1\,9}$	$\tau_{1\,10}$	$\tau_{1\,11}$	$\tau_{1\,12}$	$\tau_{1\,13}$
C_{ij}^{w}	5	3	6	6	6	3	6	6	6	8	2	8	2
$Prio_{ij}$	220	219	100	200	100	215	200	100	200	211	210	209	208
R_{ij}^{w}	2455	4908	6184	9070	7394	11523	9070	12766	14409	13981	**16433**	16867	**19319**
Functionality	**RBC Communication-session establishment - RBC-CS**												
τ_{ij}	-	$\tau_{1\,14}$	$\tau_{1\,15}$	$\tau_{1\,16}$	$\tau_{1\,17}$	$\tau_{1\,18}$	$\tau_{1\,19}$	$\tau_{1\,20}$	$\tau_{1\,21}$	$\tau_{1\,22}$	$\tau_{1\,23}$	$\tau_{1\,24}$	$\tau_{1\,25}$
C_{ij}^{w}	-	15	6	6	6	15	6	6	6	40	10	40	10
$Prio_{ij}$	-	119	100	200	100	115	200	100	200	111	110	109	8
R_{ij}^{w}	-	4933	6184	9070	7413	11548	9070	12785	14422	14040	**16500**	16914	**24374**
Functionality	**Parameter visualization - PV-DMI**												
τ_{ij}	-	$\tau_{1\,26}$	$\tau_{1\,27}$	$\tau_{1\,28}$	$\tau_{1\,29}$	$\tau_{1\,30}$	$\tau_{1\,31}$	$\tau_{1\,32}$	$\tau_{1\,33}$	$\tau_{1\,34}$	$\tau_{1\,35}$	$\tau_{1\,36}$	$\tau_{1\,37}$
C_{ij}^{w}	-	30	6	6	6	30	6	6	6	80	20	80	20
$Prio_{ij}$	-	19	100	200	100	15	200	100	200	11	10	9	8
R_{ij}^{w}	-	7453	6184	9070	8702	16528	10333	17759	19390	21508	**23978**	24370	**26840**

model by replicating the sequence of steps following it a number of times equal to the number of predecessor steps [9]. We have updated MAST by integrating Equations 3 and 4 in its offset analysis tools, in order to allow the application of these techniques to multipath e2e flows for FP scheduling. Until now, the analysis of these multipath e2e flows for the FP scheduling policy was only supported under the holistic technique, which is the most pessimistic for distributed systems. The offset- based techniques included in MAST that can now be applied to multipath flows are: the offset-based approximate analysis [11], the offset-based slanted analysis [32] and the offset-based brute force analysis [33].

VI. INDUSTRIAL USE-CASE EVALUATION

In this section, we evaluate the proposed extensions to the response-time analysis of multipath end-to-end flows. Since this work has been motivated by a real industrial need, the use case described in Section II and modeled in Section V is going to be analyzed by applying the prototype tool.

As said in Section II, the use-case functionalities have their deadlines imposed by the standard: for the three functionalities under analysis in this work, deadlines are all 1s [34]. According to the use case model presented in section IV, steps $\tau_{1\,11}$, $\tau_{1\,13}$, $\tau_{1\,23}$, $\tau_{1\,25}$, $\tau_{1\,35}$ and $\tau_{1\,37}$ are the ones that must meet these deadlines.

The railway manufacturer provided average and worst-case execution times for each function measured in the current application for the EB functionality, although these values cannot be shown due to confidentiality reasons. Based on these measures, we choose worst-case execution times for each step which are in the same magnitude order as the real ones. Best-case execution times have been assumed to be 50% of the worst-case ones. Furthermore, it is common that different functionalities have different workloads, and to highlight this effect we assume that the execution times of steps in RBC-CS are five times higher than the EB ones, and execution times of the the PV-DMI steps are 10 times higher. All these values are shown in Table 3.

Priorities are assigned to each step in the range 1..255 and are valid in the context of each partition. For the processing partition (P1 in each processor), the assignment is as follows: the EB functionality is considered to be the highest priority functionality among all, followed by the RBC-CS functionality and finally the PV-DMI. Within each functionality, priorities of the steps related to processing tasks are assigned following a decreasing order. In the I/O partition (P2 in each processor), we will only have two tasks for all the functionalities: the receiving task with a higher priority than the sending task. This is a common way to implement communication drivers, and from the modeling point of view, this means that all the steps for sending are executed at the same priority (and similarly for receiving). Values of priorities are also shown in Table 3.

The application in use is based on a cyclic executive, and our intention is to explore the possibility of using time-partitioning with a twofold objective: (1) to keep the application with the required SIL by guaranteeing that the time-requirements are met, and (2) to enable the use of the remaining capacity for other application components or even other applications. So, a tentative partition configuration that makes use of a small processing capacity, enough for the execution requirements, has been proposed. The optimization of partitioning as well as the priority assignment are beyond the scope of this work, and will be left for future work.

Partitions are allocated within a 10000μs MAF. P1 is composed of 50μs windows every 2500μs, that is: $Win_{1\,1} = \{0, 50\}$, $Win_{1\,2} = \{2500, 50\}$, $Win_{1\,3} = \{5000, 50\}$ and $Win_{1\,4} = \{7500, 50\}$. P2 is defined by 25$\mu$s windows executed every 1250μs: $Win_{2\,1} = \{1000, 25\}$, $Win_{2\,2} = \{2250, 25\}$, $Win_{2\,3} = \{3500, 25\}$, $Win_{2\,4} = \{4750, 25\}$, $Win_{2\,5} = \{6000, 25\}$, $Win_{2\,6} = \{7250, 25\}$, $Win_{2\,7} = \{8500, 25\}$ and $Win_{2\,8} = \{9750, 25\}$. We do not consider context switch times, as measurements performed in state-of-the-art processors give context switch overheads of a few microseconds, which are negligible compared to the partition lengths. As said in Section III, this work does not consider a specific network, so it is modeled as a black box producing a bounded latency that can can be measured in different ways, so this model is still realistic. For instance, if the real-time communications network would be an AFDX network [35], the work in [36] shows how to integrate the analysis of the messages in the network into the composite response

TABLE 4. Synthetic application analysis (times in ms).

e2e flow 0	Period=1157.14											
$Step$	0	1	2	3	4	5	6	7	8	9	10	11
$Processor$	1	2	3	4	1	2	3	4	1	2	2	4
C_{ij}^{w}	49.422	52.8329	60.6002	38.0098	63.6503	51.1754	43.1472	60.6002	49.422	48.7003	35.5831	53.6476
$Prio_{ij}$	23	22	21	20	19	18	17	16	15	14	13	12
R_{ij}^{w} Holistic	246.41	496.03	733.67	888.8	1024.18	1019.07	1287.14	1259.99	1443.7	**1703.59**	1991.2	1691.96
R_{ij}^{w} Offset-based	208.98	418.57	626.44	781.57	882.23	834.93	1031.17	1005.12	1043.91	1350.71	**1507.12**	1196.72
$Step$	12	13	14	15	16	17	18	19	20	21	22	-
$Processor$	1	3	3	4	1	2	3	4	1	2	3	-
C_{ij}^{w}	63.6503	55.664	60.6002	51.1754	38.8143	38.8143	38.8143	41.479	35.5831	57.5469	51.1754	-
$Prio_{ij}$	11	10	9	8	7	6	5	4	3	2	1	-
R_{ij}^{w} Holistic	1609.05	2464.51	2584.14	2087.08	2348.47	**3184.57**	**2918.61**	2974.18	3683.62	**4379.16**	**4661.58**	-
R_{ij}^{w} Offset-based	1138.24	1813.96	2005.45	1452.37	1653.41	**2305.25**	**2045.7**	2053.73	2657.43	**3157.48**	**3286.54**	-
e2e flow 1	**Period=385.714**											
$Step$	0	1	2	3	4	5	6	7	-	-	-	-
$Processor$	1	2	3	3	1	2	4	4	-	-	-	-
C_{ij}^{w}	43.0974	38.8143	53.6476	53.6476	63.0805	41.479	41.479	38.8143	-	-	-	-
$Prio_{ij}$	42	41	40	39	38	37	36	35	-	-	-	-
R_{ij}^{w} Holistic	30.39	57.94	95.37	132.8	**169.49**	152.23	**124.69**	**209.09**	-	-	-	-
R_{ij}^{w} Offset-based	30.39	57.94	95.37	132.8	**139.09**	124.69	**124.69**	**160.34**	-	-	-	-
e2e flow 2	**Period=540**											
$Step$	0	1	2	3	4	5	6	7	8	9	10	-
$Processor$	1	2	4	3	1	2	3	4	1	2	3	-
C_{ij}^{w}	53.6476	51.1754	38.8143	35.5831	35.5831	57.5469	51.1754	63.0805	53.6476	57.5469	42.1517	-
$Prio_{ij}$	34	33	32	31	30	29	28	27	26	25	24	-
R_{ij}^{w} Holistic	111.55	204.19	288.59	**211.8**	425.53	421.26	424.62	**552.74**	**598.99**	**597.32**	**591.33**	-
R_{ij}^{w} Offset-based	111.54	204.19	288.59	**211.8**	388.10	385.49	399.24	**499.82**	**510.79**	**496.13**	**492.73**	-

time analysis technique described in Section V. In order to show the effects of the network in the analysis we assume, based on our experience, that the maximum latencies for all messages are 400 μs and the minimum are 40 μs, considering a low loaded 100Mb/s switched Ethernet network and 500Mb average message sizes.

Finally, Table 3 also shows the worst-case response times obtained by the analysis. As it can be seen, deadlines are met comfortably, but the main promising fact is that these response-times have been guaranteed with a partition configuration proposal that uses only 4% of the CPU. Other real-time applications allocated in other partitions could be analyzed separately, and in case any of them would not meet its deadlines, the results from the analysis could be used to reconfigure the partitioning scheme.

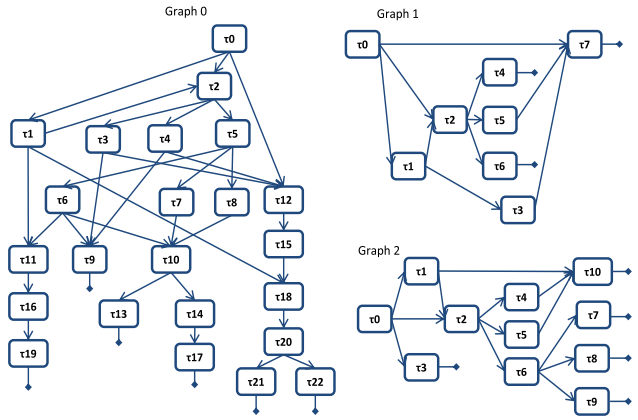


FIGURE 5. Synthetic application generated with TGFF.

VII. RESPONSE-TIME ANALYSIS PERFORMANCE

With the aim of completing the assessment on this new technique focusing now on general distributed systems scheduled by a FP policy, we are going to analyze different logical sequences of steps by generating a synthetic application using the tool Task Graphs For Free (TGFF) [37]. The DAG model employed in TGFF can be directly implemented in MAST and also in our prototype tool as all the needed elements are available. Thanks to the contributions of this work, we are capable of comparing for the first time the results on worst-case response times of multipath flows between the holistic analysis [9] and the offset-based slanted analysis [32]. We are committed to ensure that all experiments in this paper are fully replicable by the real-time community, that is why we focus in concrete instances of analysis problems rather than providing massive results based on generic values.

We have designed an architecture composed by three multipath e2e flows and four processors. For the seek of simplicity network connections, which can be modeled and analyzed along with processors, have not been included in this experiment. Each e2e flow has a different activation rate, and there may be more than one output and hence several deadline requirements. Step-to-processor mapping is made randomly, being the only restriction that two consecutive steps are allocated to different processors. Steps' worst-case execution times are generated randomly in the range of 25-45ms, and best-case execution times are assumed to be half of the worst-case ones. Priorities are assigned in a decreasing manner within each flow, and flows are prioritized with regard to their periods; the lower their period is the higher are the priorities assigned to their steps. These decisions are taken in order to avoid assigning the same priorities to different steps,

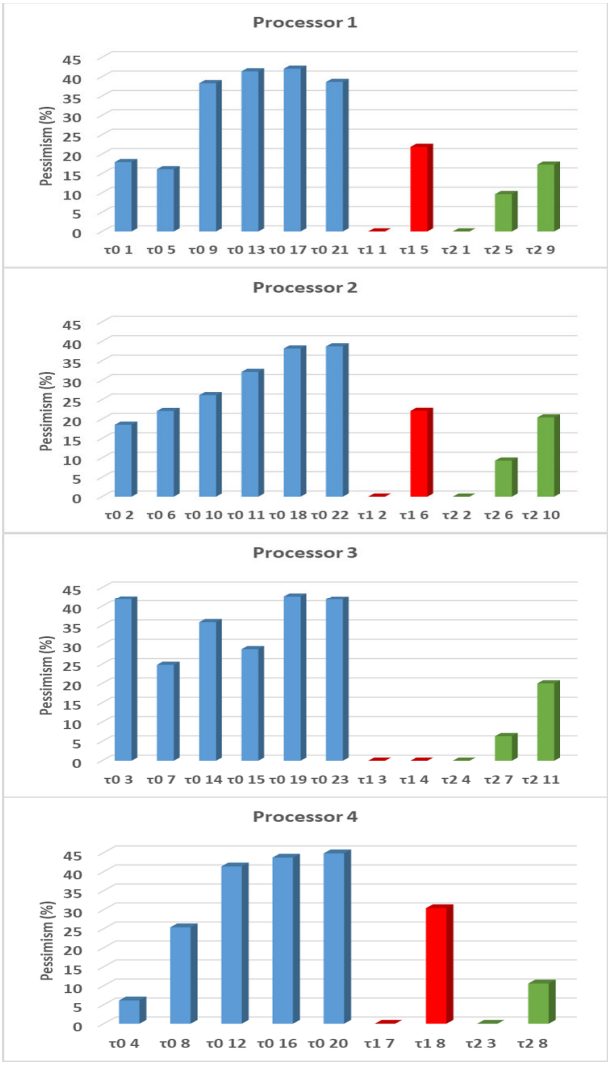


FIGURE 6. Differences between response times obtained with the holistic and the offset-based analysis.

and bearing in mind that priority assignment and optimization is beyond the scope of this work. The complete configuration of the experiment is shown in Table 4. Results for both holistic and offset-based analysis are shown too: values corresponding to those steps with deadline requirements are highlighted in bold.

These are the input parameters for the graph generation (readers are encouraged to visit the reference on this tool for a deeper understanding on DAG generation), and the generated flows have been depicted in figure 5:

```
tg_cnt 3
task_cnt 15 2
task_degree 3 3
period_mul 0.5,0.7,1.5,1,0.3,5
tg_write
eps_write

table_cnt 1
type_attrib exec_time 35 10
trans\_write
```

It is stated in Section VI that the industrial use-case that has motivated this work represents a very low loaded system. That is why now we choose the evaluation of an application that makes use of a higher percentage of the available CPU time (an average of 51%), so that we complete the experimental evaluation of this new analysis.

As said before, the holistic analysis assumes that tasks are independent and the offset-based reduces the pessimism of this approach [11], and that is exactly what it can be observed in this evaluation: all the response-time values obtained by our new technique are equal or lower than those obtained by means of the holistic analysis. Moreover, it is common that in distributed systems deadlines are larger than the e2e flows' periods, so if in our case deadlines were three or even four times the periods, the offset-based analysis would obtain schedulable solutions while the holistic analysis would not. This pessimism has been represented in Figure 6, where the difference in percentage between the worst-case response times obtained with both analysis techniques have been plotted for each step (for the sake of clarity each flow has been plotted in a different colour). While high priority tasks may not be affected by the pessimism of the holistic analysis (a 0% of pessimism is obtained in the analysis of these tasks), low priority ones do clearly exhibit this effect as their worst-case response times are always higher than the ones obtained by our tool, showing up to 40% improvements in their worst-case response times. We therefore confirm our initial notion that the offset-based analysis applied to multipath flows would outperform the results from the holistic analysis, which was the only analysis technique available until now.

VIII. CONCLUSION AND FUTURE WORK

In this paper a new schedulability analysis method for hierarchical time partitioned distributed real-time systems has been proposed, so that timing behavior of multipath end-to-end flows can be evaluated. This is motivated by a real industrial railway application and the need to accurately analyze its timing behavior, in order to perform a complete reconfiguration on its architecture and execution model aided by this tool. It is common practice that different manufacturers take part in the system design, integrating different applications in separated partitions. With this new analysis each of these complex applications can be separately analyzed and then integrated to check the overall schedulability. Moreover, thanks to our contribution on the extension of the offset-based analysis to support multipath flows, works such as [38] from the automotive domain are no longer obliged to relax task dependencies and simplify their models in order to turn them linear.

As future work, research is going to be focused on scheduling optimization, both at the primary scheduler, via time-partition optimization, and at the secondary scheduler by proposing priority assignment techniques. Network level analysis and optimization, which has not been addressed in this work, is also an open issue for nowadays industrial real-time applications, and thanks to our black-box model

we will be able to integrate such analysis with the technique developed in this work.

REFERENCES

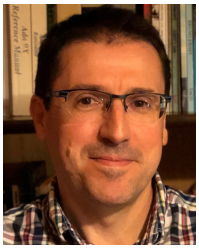
[1] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Syst. J.*, vol. 9, no. 2, pp. 350–365, Jun. 2015.
[2] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, nos. 2–3, pp. 101–155, Nov. 2004.
[3] *Avionics Application Software Standard Interface. Arinc Specification 653- 1*, Airlines Electron. Eng. Committee, Aeronaut. Radio INC, Annapolis, MD, USA, vol. 2551, 2010, pp. 21401–7435.
[4] H. Fang and R. Obermaisser, "Execution environment for mixed-criticality train applications based on an integrated architecture," in *Proc. Int. Conf. Promising Electron. Technol. (ICPET)*, Oct. 2017, pp. 1–7.
[5] J. Rushby, "Partitioning in avionics architectures: Requirements, mechanisms, and assurance," Sri Int. Menlo Park Ca Computer Science Lab, Menlo Park, CA, USA, Tech. Rep., 2000.
[6] D. J. Smith and K. G. Simpson, *Functional Safety: A straightforward guide to Applying IEC 61508 and Related Standards*, Standard IEC 61508, Routledge, 2004.
[7] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 1–44, Oct. 2011.
[8] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogramm.*, vol. 40, nos. 2–3, pp. 117–134, 1994.
[9] J. J. G. Garcia, J. C. P. Gutierrez, and M. G. Harbour, "Schedulability analysis of distributed hard real-time systems with multiple-event synchronization," in *Proc. 12th Euromicro Conf. Real-Time Syst. Euromicro RTS*, 2000, pp. 15–24.
[10] R. Garibay-Martínez, G. Nelissen, L. Lino Ferreira, P. Pedreiras, and L. Miguel Pinho, "Improved holistic analysis for Fork–Join distributed real-time tasks supported by the FTT-SE protocol," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1865–1876, Oct. 2016.
[11] J. C. Palencia and M. Gonzalez Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proc. 19th IEEE Real-Time Syst. Symp.*, Dec. 1998, pp. 26–37.
[12] J. C. Palencia, M. G. Harbour, J. J. Gutierrez, and J. M. Rivas, "Response-time analysis in hierarchically-scheduled time-partitioned distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2017–2030, Jul. 2017.
[13] S. O. Marinescu, D. Tamas-Selicean, V. Acretoaie, and P. Pop, "Timing analysis of mixed-criticality hard real-time applications implemented on distributed partitioned architectures," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2012, pp. 1–4.
[14] J. Li, D. Ferry, S. Ahuja, K. Agrawal, C. Gill, and C. Lu, "Mixed-criticality federated scheduling for parallel real-time tasks," *Real-Time Syst.*, vol. 53, no. 5, pp. 760–811, Sep. 2017.
[15] M. W. Anwar, M. Rashid, F. Azam, A. Naeem, M. Kashif, and W. H. Butt, "A unified model-based framework for the simplified execution of static and dynamic assertion-based verification," *IEEE Access*, vol. 8, pp. 104407–104431, 2020.
[16] M. W. Anwar, M. Rashid, F. Azam, M. Kashif, and W. H. Butt, "A model-driven framework for design and verification of embedded systems through systemverilog," *Design Autom. Embedded Syst.*, vol. 23, nos. 3–4, pp. 179–223, Dec. 2019.
[17] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.
[18] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou, "Testing real-time systems using UPPAAL," in *Formal Methods and Testing*. Cham, Switzerland: Springer, 2008, pp. 77–117.
[19] P. Han, Z. Zhai, and L. Zhang, "A model-based approach to optimizing partition scheduling of integrated modular avionics systems," *Electronics*, vol. 9, no. 8, p. 1281, Aug. 2020.
[20] S. Perathoner, E. Wandeler, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst, and M. González Harbour, "Influence of different abstractions on the performance analysis of distributed hard real-time systems," *Design Autom. Embedded Syst.*, vol. 13, nos. 1–2, pp. 27–49, Jun. 2009.
[21] *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems Part 1: General Requirements*, Standard IEC 61508, 2010.
[22] M. Gonzalez Harbour, J. J. Gutierrez Garcia, J. C. Palencia Gutierrez, and J. M. Drake Moyano, "MAST: Modeling and analysis suite for real time applications," in *Proc. 13th Euromicro Conf. Real-Time Syst.*, 2001, pp. 125–134.
[23] *UML Profile for MARTE: Modeling and Analysis of Real Time Embedded Systems, Version 1.1*, Object Management Group, OMG Document Formal, 2011.
[24] ERTMS/ETCS, "European Rail Traffic Management System/European Train Control System release notes to system requirements specification," *Subset 026 version 2.3.0*, 2006.
[25] *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems Part 6: Guidelines on the Application of IEC 61508-2 and IEC 61508-3*, Stanadrd IEC 61508, 2010.
[26] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surv.*, vol. 50, no. 6, p. 82, 2018.
[27] D. Reinhardt and G. Morgan, "An embedded hypervisor for safety-relevant automotive E/E-systems," in *Proc. 9th IEEE Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2014, pp. 189–198.
[28] H. Herpel, A. Schuettauf, G. Willich, S. Tverdyshev, S. Pletner, F. Schoen, B. Kiewe, C. Fidi, M. Maeke-Kail, and K. Eckstein, "Open modular computing platforms in space—Learning from other industrial domains," in *Proc. IEEE Aerosp. Conf.*, Mar. 2016, pp. 1–11.
[29] J. M. Rivas, J. J. Gutierrez, J. C. Palencia, and M. G. Harbour, "Schedulability analysis and optimization of heterogeneous EDF and FP distributed real-time systems," in *Proc. 23rd Euromicro Conf. Real-Time Syst.*, Jul. 2011, pp. 195–204.
[30] J. M. Rivas, J. J. Gutiérrez, M. Aldea, C. Cuevas, M. G. Harbour, J. M. Drake, J. L. Medina, L. Rioux, R. Henia, and N. Sordon, "An experience integrating response-time analysis and optimization with an MDE strategy," in *Proc. Fed. Int. Conf. Softw. Technol., Appl. Found.* Cham, Switzerland: Springer, 2016, pp. 303–316.
[31] M. González Harbour, J. J. Gutiérrez, J. M. Drake, P. López Martínez, and J. C. Palencia, "Modeling distributed real-time systems with MAST 2," *J. Syst. Archit.*, vol. 59, no. 6, pp. 331–340, Jun. 2013.
[32] J. Mäki-Turja and M. Nolin, "Efficient implementation of tight response-times for tasks with offsets," *Real-Time Syst.*, vol. 40, no. 1, pp. 77–116, Oct. 2008.
[33] K. Tindell, "Adding time-offsets to schedulability analysis," Dept. Comput. Sci., Univ. York, York, U.K., Tech. Rep. YCS 221, Jan. 1994.
[34] *ERTMS/ETCS—SUBSET-041: Performance Requirements for Interoperability*, UNISIG Standard SUBSET-041, 2015.
[35] *Arinc Specification 664P7: Aircraft Data Network, Part 7–Avionics Full Duplex Switched Ethernet (AFDX) Network*, Aeronaut. Radio, Inc., Annapolis, MD, USA, vol. 2551, 2009, pp. 21401–7435.
[36] J. J. Gutiérrez, J. C. Palencia, and M. González Harbour, "Holistic schedulability analysis for multipacket messages in AFDX networks," *Real-Time Syst.*, vol. 50, no. 2, pp. 230–269, Mar. 2014.
[37] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. 6th Int. Workshop Hardw./Softw. Codesign. (CODES/CASHE)*, 1998, pp. 97–101.
[38] S. Anssi, S. Kuntz, S. Gérard, and F. Terrier, "On the gap between schedulability tests and an automotive task model," *J. Syst. Archit.*, vol. 59, no. 6, pp. 341–350, Jun. 2013.

ANDONI AMURRIO received the B.S. degree in telecommunications systems engineering from the University of the Basque Country, Bilbao, in 2014, and the M.S. degree in telecommunications engineering from the Autonomous University of Barcelona, Barcelona, in 2016. He is currently pursuing the Ph.D. degree in science and technology with the IKERLAN Research Center, in collaboration with the University of Cantabria.

He spent some years working as a Network Technician for Telefónica Movistar and later as an Engineer for SEAT, S.A. His research interests include the schedualbility analysis and scheduling algorithms for time-partitioned distributed real-time systems.

EKAIN AZKETA received the B.S. and M.S. degrees in telecommunications engineering from Mondragon University, Mondragon, Spain, in 2004 and 2007, respectively, and the Ph.D. degree in telecommunications engineering from the University of the Basque Country, Bilbao, Spain, in 2013.

From 2007 to 2013, he was a Research Assistant with the Department of Software Technologies, IKERLAN Technology Research Center, Mondragon. Since 2013, he has been a Researcher with the IKERLAN Technology Research Center, where he is currently with the Industrial Cybersecurity Team. His research interests include the optimization techniques for deployment and scheduling of distributed real-time systems, and architectures for mixed-criticality systems.



J. JAVIER GUTIERREZ received the B.S. and Ph.D. degrees from the University of Cantabria, Spain, in 1989 and 1995, respectively.

He has been an Associate Professor with the Software Engineering and Real-Time Group, University of Cantabria, since 1996, where he works in software engineering for real-time. His research activity deals with the scheduling, analysis, and optimization of embedded real-time distributed systems. He has been involved in several research projects building real-time controllers for robots, evaluating Ada for real-time applications, developing middleware for real-time distributed systems, and proposing models along with the analysis and optimization techniques for distributed real-time applications.

MARIO ALDEA is currently an Associate Professor with the Department of Electronics and Computers, University of Cantabria, Spain. His research interests include real-time systems, with special focus on flexible scheduling, real-time operating systems, and real-time languages. He has been involved in several research and industrial projects related with real-time and embedded technologies. He is also the main developer of MaRTE OS, an operating system that has served as platform to provide support for advanced real-time services.

MICHAEL GONZÁLEZ HARBOUR is currently a Professor with the Department of Computer Science and Electronics, University of Cantabria. He works in software engineering for real-time systems, and particularly in modeling and schedulability analysis of distributed real-time systems, real-time operating systems, and real-time languages. He is the coauthor of *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. He has been involved in several industrial projects using Ada to build real-time controllers for robots. He has participated in the real-time working group of the POSIX standard for portable operating system interfaces. He is one of the principal authors of the MAST suite for modeling and analysing real-time systems.