applied sciences

MDPI

Article

Schedulability Analysis in Fixed-Priority Real-Time Multicore Systems with Contention

Luis Ortiz †, Ana Guasque *,†, Patricia Balbastre †, José Simó † and Alfons Crespo †

Instituto de Automática e Informática Industrial (ai2), Universitat Politècnica de València, 46022 Valencia, Spain; luioren@ai2.upv.es (L.O.); patricia@ai2.upv.es (P.B.); jsimo@ai2.upv.es (J.S.); acrespo@ai2.upv.es (A.C.)

* Correspondence: anguaor@ai2.upv.es

† All authors contributed equally to this work.

Abstract: In the scheduling of hard real-time systems on multicore platforms, significant unpredictability arises from interference caused by shared hardware resources. The objective of this paper is to offer a schedulability analysis for such systems by assuming a general model that introduces interference as a time parameter for each task. The analysis assumes constrained deadlines and is provided for fixed priorities. It is based on worst-case response time analysis, which exists in the literature for monocore systems. We demonstrate that the worst-case response time is an upper bound, and we evaluate our proposal with synthetic loads and execution on a real platform.

Keywords: static scheduling; real-time systems; partitioned systems; multicore systems; worst-case response time

1. Introduction

Multicore platforms are growing in importance in critical systems, such as automotive or avionic applications. In these systems, failure to meet any requirement may lead to catastrophic consequences. Therefore, compliance with deadlines in hard real-time multicore systems is a crucial aspect.

The scheduling problem in multicore systems in turn requires solving two problems: (1) decide which tasks are executed in each core (allocation problem) and (2) execute each task to meet all the deadlines (scheduling problem). In partitioned multicore systems where migration is not allowed, each core can be executed independently, so that monocore scheduling theory can be applied. But this is not possible when shared hardware resources exist. When a task runs in parallel with other tasks in other cores, it may lead to delays due to shared hardware resources, such as caches, buses, and memories [1,2]. In this case, interference among cores is produced, and it causes unpredictability, which is undesired in critical systems.

Modeling interference is not an easy task, and it has been a hot topic in recent years. There are two main trends to take into account for interference in the temporal model. On the one hand, there are methods that focus on the modeling and interference mitigation of specific hardware resources [3]. These methods are only valid for this type of resources, but they are able to adjust the interference produced in a very accurate way. On the other hand, other methods analyze contention for multiple resources and their integration in schedulability analysis. When multiple shared resources are considered, the contention model must cope with any of the resources, so a general contention model is required [4]. The drawback is that the interference produced is overestimated, but sometimes, it is the only feasible approach when the chip vendor does not provide details about hardware behavior.

In addition to modeling interference, it is also necessary to act in both the allocation and scheduling stages to reduce the overall interference produced. The allocation problem is recognized to be NP-hard in the strictest sense [5]. In multicore systems, the way tasks are

check for updates

Citation: Ortiz, L.; Guasque, A.; Balbastre, P.; Simó, J.; Crespo, A. Schedulability Analysis in Fixed-Priority Real-Time Multicore Systems with Contention. *Appl. Sci.* **2024**, *14*, 4033. https://doi.org/10.3390/app14104033

Received: 18 March 2024
Revised: 22 April 2024
Accepted: 7 May 2024
Published: 9 May 2024

allocated to cores largely determines the global interference [6]. Some of the main allocation techniques are heuristics based on bin packing, such as First Fit, Worst Fit [7,8], etc. These techniques consist of mapping tasks to cores according to their loads (or utilization), without considering interference. But the allocation strategy Wmin, presented in [4], reduces overall interference, as it aims to group tasks with interference in the same core, whenever possible. This work assumes a general model for interference where this parameter is added to the traditional parameters of a task.

If the allocation strategy greatly reduces interference, as is the case with Wmin, traditional scheduling algorithms such as fixed or dynamic priorities can be adopted in the scheduling phase. Because the temporal task model is modified by the inclusion of interference due to contention, it is necessary to provide new schedulability tests for these models. The work presented in [9] presents schedulability tests for dynamic priorities under the assumption of the temporal model presented in [4].

When it comes to schedulability tests for fixed priorities, they are based on worst-case response time (WCRT). WCRT is the length of the longest interval from a task's release till its completion [10]. These schedulability tests do not consider interference, so this work fills this gap. Furthermore, when interference is taken into account, the worst-case scenario may occur at any point in time during the entire execution window, not necessarily upon the first activation of the task [9]. Extending the concept of WCRT to consider this is also challenging.

Regarding scheduling strategies to reduce interference, the recent work [6] proposes the Rolling Horizon MILP Algorithm (RHMA), which not only achieves a feasible schedule but also minimizes the interference generated by shared hardware resources within the context of hard real-time multicore systems.

Contribution

This paper proposes a schedulability test for fixed priorities in hard real-time multicore systems. We assume the temporal model defined in [4], which considers the interference generated by shared hardware resources. We extend the work presented in [9] to consider fixed priorities in hard real-time systems. In particular, we propose a schedulability test based on WCRT. The novelty of the contribution is that both the task model and worst-case response time equation incorporate interference due to shared hardware resources.

The rest of the paper is organized as follows: Section 2 presents the main contributions in the related research area. In Section 3, we present the system model for constrained-deadline systems that is used throughout this paper. Section 4 reviews the classical schedulability analysis for multicore systems with fixed priorities. Then, we propose an upper bound for the classical worst-case response time analysis, considering contention due to shared hardware resources. We also propose the corresponding schedulability test. The experimental evaluation in Section 5 exposes the compliance of our schedulability test. It also assesses different allocation techniques by comparing how closely the suggested algorithm matches the real value. These results are confirmed by a real case. Finally, Section 6 presents the main conclusions and further lines of research.

2. Related Works

The evolution from single-core to multicore computing systems has been a gradual process. Beginning with initial investigations into hard real-time multicore scheduling in the late 1960s, the field has seen a steady stream of research developments aimed at adapting traditional single-core scheduling techniques to multicore environments. Various approaches have been proposed and explored over the years, each contributing to the evolving landscape of multicore scheduling methodologies.

A comprehensive survey of the existing literature on multicore scheduling, as compiled in [11], serves as a valuable resource for understanding the breadth and depth of research in this area. This compilation encapsulates the spectrum of techniques and findings relevant to multicore scheduling, providing insights into the progress made and the challenges that lie ahead.

In recent years, there has been a surge in research focused on reducing interference in multicore systems. A notable contribution to this field is the recent survey by Lugo et al. [12], which highlights key advancements in interference reduction techniques within multicore systems over the past five years. This survey provides a comprehensive overview of the latest research efforts and their contributions to mitigating interference in multicore environments.

Various interference models in the literature target specific sources of interference within computer systems. Some models concentrate on interference stemming from the main memory [13,14], while others focus on interference generated by cache memory [15,16], and others yet address interference originating from the memory bus [17,18]. Each of these models incorporates the impacts of interference into schedulability analysis, typically focusing on a single shared hardware resource at a time.

Some of the works that consider a general model and are closer to our work are detailed hereunder. Altmeyer et al. [19] presented the Multicore Response Time Analysis (MRTA) framework, designed to integrate task demands on various shared resources with the resources' supply and to incorporate the resulting interference explicitly into response time analysis. Notably, they eschewed the concept of worst-case execution time (WCET) and instead assumed fixed-priority preemptive scheduling.

The research described in [20] introduced a response time analysis method tailored for real-time partitioned multiprocessor systems. These systems consist of a set of independent tasks with fixed priorities, which can be activated arbitrarily and share secondary resources. In the above system model, the worst-case response time (WCRT) of a task may exceed its period, allowing the task to potentially re-arrive before the completion of its preceding job. Additionally, both local and global shared resources are accounted for, and a cap on the maximum number of requests any task can make to a shared resource is imposed.

The study outlined in [21] adopted the concept of superblocks to represent real-time tasks. The WCRT of tasks is deduced from the worst-case completion time of a superblock, determined by the upper bound on access requests to shared memory and the maximum required computation time. The authors investigated various hardware access models and their impact on schedulability.

Choi et al. [22] presented a response time analysis method tailored for synchronous dataflow programs mapped to multiple parallel dependent tasks executing on a computer cluster consisting of many cores. This technique extends the MRTA framework [19] to suit a specific architecture. The WCRT analysis module determines the upper limit of demand for shared resources based on the event stream model of each cluster in each processing element.

The authors in [23] examined fixed-priority partitioned multiprocessor scheduling. They demonstrated that for task systems with arbitrary deadlines, a greedy mapping strategy achieves a speedup factor that holds true for both polynomial-time and exponential-time schedulability tests. They provided schedulability tests specifically tailored for Deadline Monotonic (DM) scheduling, encompassing implicit, constrained, or arbitrary deadline models. However, none of these tests incorporate shared hardware resources into their definitions.

Chen et al. [24] introduced a schedulability condition initially designed for uniprocessor systems, later extending it to derive an upper limit on the number of cores necessary for a periodic, non-preemptive task set. Additionally, they proposed a comprehensive formula to determine valid start time offsets for a new task to collaborate with existing tasks on the same processor. Consequently, offsets and non-preemption are incorporated, which deviates from our temporal model, and the proposed schedulability condition overlooks interference considerations.

Huang et al. [25] introduced an asymmetric analysis method for evaluating the response time of task processing and access to shared resources, considering both the execution and suspension intervals of tasks. Their model incorporates an upper limit on the execution time of shared resource access for each task and the maximum count of segments for accessing resources per task, with each segment potentially comprising multiple consecutive data requests to access the shared resource. In contrast, our approach offers greater flexibility by not constraining the amount of interference or the number of access times in the model.

Choi et al. [26] proposed a fundamental technique for estimating the WCRT of synchronous dataflow (SDF) applications in multicore systems, accounting for task dependency and execution time variations. Their approach integrates schedule time bound analysis, which accounts for interference between task instances within the same SDF graph, and response time analysis, which addresses interference from other real-time tasks.

While both studies consider interference from other real-time tasks in response time analysis, their work focuses on SDFs, whereas ours does not involve dataflows. In [27], the allocation problem and exact computation of WCRT are addressed by using a compositional framework based on time automata.

Andersson et al. [28] introduced a schedulability test for the fixed-priority preemptive partitioned scheduling of sporadic, constrained-deadline tasks. Their approach accounts for varying the execution times of tasks based on specific co-runners, which are determined through static analysis or measurements. In a similar vein, Al-Bayati et al. [29] concentrated on resource-aware partitioning approaches within the context of fixed-priority partitioned scheduling. They proposed an Integer Linear Programming formulation to efficiently assign tasks to cores, assign priorities to the tasks, and select resource protection mechanisms.

This work follows the general system model in [4], which introduced a scheduling algorithm that precisely computes total interference and an allocator that aims to minimize this interference. However, in the above work, the authors did not propose any schedulability test to ensure the system's feasibility. For this reason, [9] proposed two schedulability tests for dynamic-priority, real-time systems with contention. Therefore, our work covers the gap of the schedulability test in [4], but it is focused on fixed-priority, real-time systems with contention.

3. System Task Model

We assume the same task model as the one presented in [4]. We suppose a multicore system with m homogeneous cores $(M_0, M_1, M_2, ..., M_{m-1})$ in which a task set τ of n independent periodic or sporadic tasks must be statically allocated. Each task τ_i is represented by the tuple

$$\tau_i = (C_i, D_i, T_i, I_i) \tag{1}$$

where C_i is the WCET, D_i is the relative deadline, T_i is the period, and I_i is the interference. We assume constrained deadlines, so $D_i \leq T_i$.

The I_i parameter was first introduced in [4]; then, it was also used in [9]. Briefly, interference parameter I_i is the worst-case time that it takes τ_i to access hardware resources and means a "delay" in other tasks executing at the same time on all other cores due to contention. I_i is the worst-case time that τ_i spends reading and writing memory operations. This is depicted in Figure 1, where we can see that all the time that τ_0 dedicates to these operations is included in interference parameter I_0. Task execution times are represented by solid rectangles while interference is sketched in dashed rectangles. Considering the viewpoint of other tasks, I_i is the additional time that τ_i causes in other tasks concurrently executing on other cores due to contention. For a more comprehensive understanding of the interference parameter, please refer to the aforementioned articles.

Although our work introduces the interference parameter in the general model, there are two commonly used approaches for modeling interference: one involves utilizing a model tailored to the particular type of shared hardware, while the other proposes a general model that is applicable to any type of hardware. The former approach provides a more accurate estimation of interference but is limited to the specific hardware for which it was developed. Additionally, this interference value is typically incorporated into WCET, resulting in a highly pessimistic solution. On the other hand, the latter approach yields a higher interference value, but by independently incorporating this parameter into the temporal model, it is possible to derive a less pessimistic model.

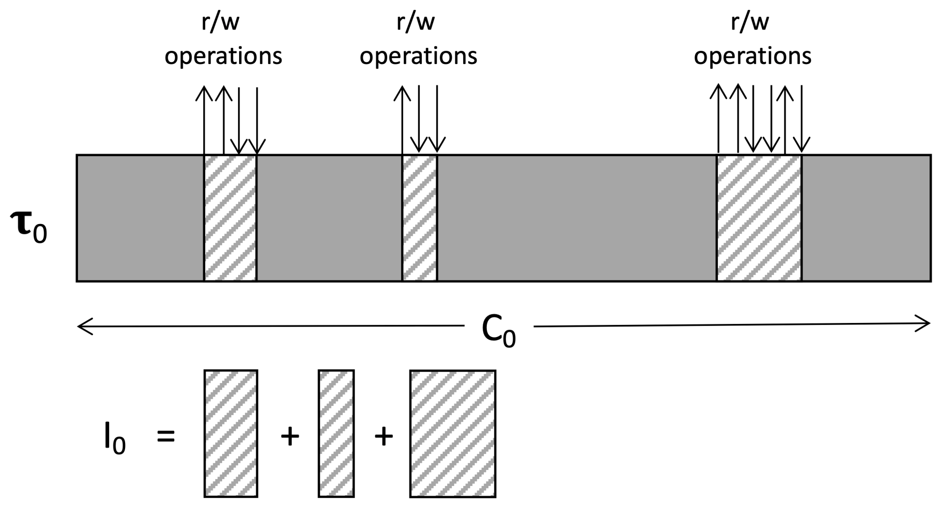

Figure 1. Example of task interference.

The hyperperiod of τ, H, is the time after which the pattern of job release times starts to repeat and is the least common multiple of the periods of all the tasks in that set. N_i is the number of activations that task τ_i has throughout the hyperperiod ($N_i = H/T_i$).

Throughout this text, we talk about tasks that belong or not to a core. When we refer to M_{τ_i}, we mean the core in which τ_i is allocated. Moreover, we denote by τ_{M_k} the set of tasks in τ that belong to core M_k. Therefore, $\bigcup_{k=0}^{m-1} \tau_{M_k} = \tau$.

Also from [4], a task is designated as a receiving task when it accesses shared hardware resources, resulting in an increase in its computation time caused by interference from other tasks allocated to different cores. Conversely, a task is classified as a broadcasting task when its access to shared hardware resources triggers an increment in computation time in other tasks allocated to different cores due to contention. If the interference parameter of task τ_i is equal to zero ($I_i = 0$), then τ_i is neither a broadcasting nor a receiving task. If $I_i > 0$ and there is at least one task τ_j in another core whose $I_j > 0$, τ_i is a broadcasting and receiving task.

Once the model is defined, let us introduce the concept of interference in the schedulability analysis of real-time multicore systems based on fixed priorities.

4. Interference-Aware Schedulability Analysis for Fixed Priorities

In this section, first, we present a well-known fixed-priority schedulability test for constrained-deadline task models. Traditional models overlook the interference arising from the concurrent execution of different tasks across multiple cores. Then, we contribute to the field in this sense by including interference in the schedulability test.

4.1. Deadline Monotonic Schedulability Analysis

Fixed-priority schedulers assign an initial priority to tasks that remains constant during all execution. Deadline Monotonic scheduling (DM) [30] considers task sets with deadlines shorter than periods and assigns the highest priority to the task with the shortest deadline. An exact schedulability test (necessary and sufficient condition) for fixed priorities is based on calculating the WCRT of each task.

The calculation of the WCRT of τ_i in a monocore system is presented in Equation (2) [31]:

$$WCRT_i = C_i + \sum_{\substack{\forall \tau_j \in hp(i) \\ \forall \tau_j \in M_{\tau_i}}} \left\lceil \frac{WCRT_i}{T_j} \right\rceil C_j \tag{2}$$

The second term of the preceding equation signifies the execution of higher-priority tasks in the worst case, that is, assuming the synchronous activation of tasks. Equation (2) is solved by an iterative method. The stop conditions are the violation of a deadline ($WCRT_i > D_i$ for any task i) or convergence ($WCRT(k+1) = WCRT(k)$).

Then, the task set is schedulable if and only if

$$WCRT_i \leq D_i \quad \forall \tau_i \in \tau \tag{3}$$

The outcome of this test not only determines whether the system is schedulable or not but also provides the worst-case response time (WCRT) of each task. Additionally, it identifies which tasks are implicated in deadline misses, if any occur.

Without interference considerations, the WCRT of all tasks in the task set is obtained when all tasks are activated simultaneously. It corresponds to the first activation of the tasks, so the response time is not calculated for all the activations but only the first.

4.2. Interference-Aware Schedulability Analysis for Deadline Monotonic Scheduling

Given the monocore classical test presented above, this section proposes a schedulability test that extends this test to multicore systems, considering the interference delay due to contention.

First, we present an upper bound of WCRT, when interference is considered. This is an upper bound because it assumes that the maximum interference is always produced. Then, we provide the corresponding schedulability test. As we define an upper bound of the response time, the test that we provide is sufficient but not necessary.

Let us start with some considerations to be taken into account. When we extend the analysis to multiple cores, would it be sufficient to consider that the worst-case response time of a task only depends on the computation times of higher-priority tasks of its core? In multicore systems with contention, the response time of a task depends on the following:

- Its own computation time.
- The interference it is affected by (if it is a receiving task).
- The computation time of higher-priority tasks allocated to its core. And this, in turn, depends on the interference they are affected by (if they are receiving tasks).

Therefore, these aspects may be considered when schedulability analysis is proposed.

As stated before, the WCRT of synchronous tasks executed in a monocore system corresponds to the first activation of the tasks. In multicore systems, when interference is considered, WCRT does not necessarily have to be relative to the first activation. Let us consider an example to report this common misconception.

Let us take task set $\tau = [\tau_0, \tau_1, \tau_2]$, with $\tau_0 = (1, 2, 3, 0)$, $\tau_1 = (2, 4, 5, 1)$, and $\tau_2 = (1, 3, 5, 1)$, allocated to a dual-core platform (Table 1). τ_0 and τ_1 are allocated to core M_0, and τ_2, to M_1. After allocating tasks to cores, the DM algorithm schedules tasks in each core. The execution chronogram of the task set is shown in Figure 2.

Table 1. System with task set τ.

Task τ	C	D	T	I	Core M
0	1	2	3	0	0
1	2	4	5	1	0
2	1	3	5	1	1

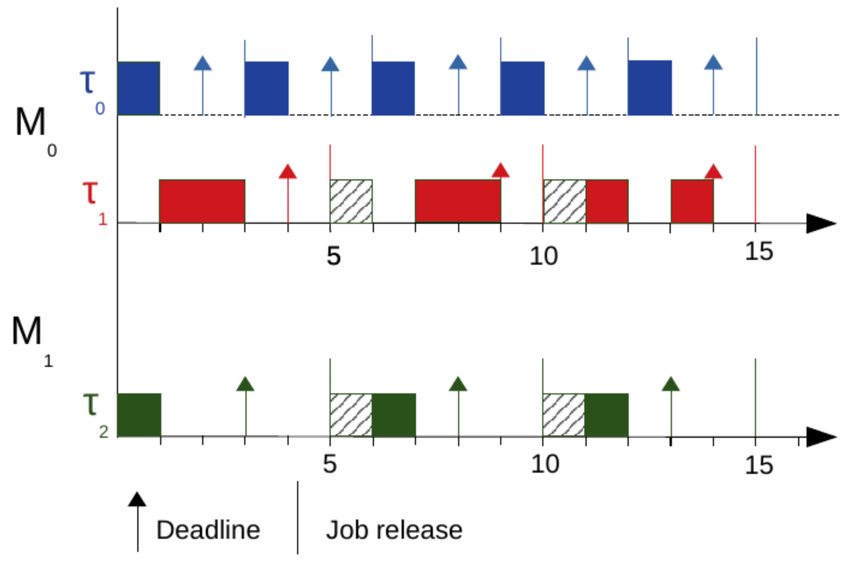

Figure 2. WCRT of tasks in multicore systems with contention.

Task τ_0 possesses the highest priority, as it is the task with the shortest deadline. As $I_0 = 0$, it is not a receiving task, so it does not receive any interference. τ_1 and τ_2 are receiving/broadcasting tasks ($I_1, I_2 > 0$). Every time τ_1 is active at the same time as τ_2 is, interference appears. As τ_0 has the highest priority, τ_1 starts its execution at time 1, when τ_2 has just finished its execution, so interference does not appear. Therefore, the response time of τ_1 at its first activation is 3. However, at time 5, τ_1 and τ_2 are released and coincide in execution. Due to this interference and its preemption due to the release of τ_0, the response time of τ_1 at its second activation is 4. The same happens at the third activation. Then, the WCRT of τ_1 does not refer to its first activation. The same happens with τ_2.

With this example, we can conclude that when interference is considered, the WCRT of a task may refer to any activation, not necessarily the first activation. For this reason, in multicore systems with contention, all activations of all tasks must be studied in order to ensure that the test is correct.

4.2.1. Previous Definitions

Below, let us derive an upper bound of the response time of all activations of all tasks allocated to a multicore system with contention. For the subsequent analysis, we need some definitions.

Definition 1. *Let $\overrightarrow{v_{j\rightarrow i}}$ be the activation pattern from a broadcasting task τ_j to a receiving task τ_i* [9].

We require this information, represented as an array, to compute the interference that a broadcasting task τ_j can induce on a receiving task τ_i. This array denotes the number of activations of τ_j that occur within an activation of τ_i. Array $\overrightarrow{v_{j\rightarrow i}}$ is calculated as a relation of periods between broadcasting and receiving tasks.

This expression is valid for implicit deadline task sets. However, if this array is used to calculate the interference between tasks when $D_i \leq T_i$, the interference obtained is highly overestimated. The reason is that there are activations of a task that fall within an activation of another that do not provoke interference because the deadline is not taken into account.

Then, let us propose an improved definition of the activation pattern for constrained deadlines, noted as $\overrightarrow{v^*_{j\rightarrow i}}$.

Definition 2. *Let $\overrightarrow{v^*_{j\rightarrow i}}$ be the improved activation pattern from a broadcasting task τ_j to a receiving task τ_i.*

This definition is similar to the one presented in [9] but considers the deadlines to count the maximum number of overlaps among activations. In this sense, if the broadcasting task is released when the deadline of the receiving task has expired and has not yet been released again (and vice versa), this overlap is not counted. Note that if τ_i is not a receiving task, $\overrightarrow{v^*_{j\to i}}[a] = 0 \quad \forall \tau_j \in \tau, \forall a \in N_i$.

We use an example to show the differences of both parameters.

Let us consider a system with two tasks, $\tau' = [\tau'_0, \tau'_1]$, with $\tau'_0 = (1,2,3,1)$ and $\tau'_1 = (1,6,7,1)$, allocated to a dual-core platform. τ'_0 is allocated to core M'_0, and τ'_1, to M'_1 (Table 2). For simplicity, computation times are not shown.

Table 2. System with task set τ'.

Task τ'	C	D	T	I	Core M
0	1	2	3	1	0
1	1	6	7	1	1

As seen in Figure 3a, $\overrightarrow{v_{1'\to 0'}} = [1,1,2,1,2,1,1]$ from [9]. Equivalently, $\overrightarrow{v_{0'\to 1'}} = [3,3,3]$. However, from Figure 3b, $\overrightarrow{v^*_{1'\to 0'}} = [1,1,1,1,1,1,1]$ and $\overrightarrow{v^*_{0'\to 1'}} = [2,3,2]$. Then, $\overrightarrow{v_{0'\to 1'}}[2] = 2$ and $\overrightarrow{v^*_{0'\to 1'}}[2] = 1$. This difference is due to the fact that first activation of τ'_1 does not overlap with the third activation of τ'_0 (considering deadlines), as D_1 just expires when τ'_0 is released. However, as $\overrightarrow{v_{1'\to 0'}}$ does not consider deadlines but periods, this overlap is counted.

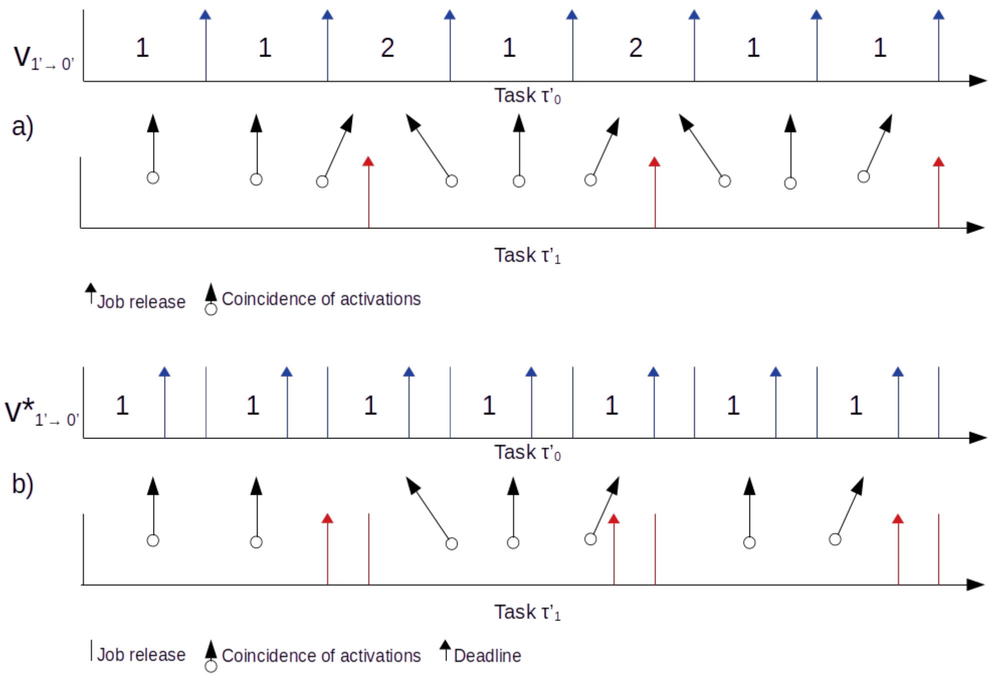

Figure 3. Example (**a**,**b**) of (a) $\overrightarrow{v_{j\to i}}$ and (b) $\overrightarrow{v^*_{j\to i}}$ for task set $\tau' = [\tau'_0, \tau'_1]$ allocated to a dual-core platform.

The next theorem proves that $\overrightarrow{v^*_{j\to i}}$ can be used to calculate an upper bound of interference.

Theorem 1. *The maximum number of activations of broadcasting task τ_j that fall within the a^{th} activation of τ_i and may provoke interference is*

$$\overrightarrow{v^*_{j\to i}}[a] = K + \sum_{t=aT_i+1}^{aT_i+D_i-1} g(t) \tag{4}$$

where

$$K = \begin{cases} 1 & \textit{If } aT_i \in [nT_j, nT_j + D_j)\ \forall n \in N_j \\ 0 & \textit{Elsewhere} \end{cases} \tag{5}$$

and

$$g(t) = \begin{cases} 1 & \textit{If } t - T_j \left\lfloor \frac{t}{T_j} \right\rfloor = 0 \\ 0 & \textit{Elsewhere} \end{cases} \tag{6}$$

Proof. Let us suppose that any a'^{th} activation of τ_i does not fall in the interval defined as $[nT_j, nT_j + D_j)$. Then, $nT_j + D_j \leq a'T_i < nT_j$. In this case, K=0, which means that the receiving task is released when the deadline of the broadcasting task has expired and that it has not yet been released again. So, there is no interference. On the contrary, if the receiving task is released when the broadcasting task has been released and its deadline has not expired, there may be interference.

Let us assume now that there exists t' so that $t' = \alpha \cdot T_j$ and $aT_i < t' < aT_i + D_i$.

In this case,

$$t' - T_j \left\lfloor \frac{t'}{T_j} \right\rfloor = \alpha \cdot T_j - T_j \left\lfloor \frac{\alpha \cdot T_j}{T_j} \right\rfloor = 0$$

thus $g(t') = 1$.

Thus, $g(t)$ equals 1 only when broadcasting task τ_j is released within the interval $(aT_i, aT_i + D_i)$. By analyzing $g(t)$ throughout the preceding interval, we determine the number of activations that occur within activation a of τ_i, considering the term "activation" as the period from the release to the deadline.

The sum of $g(t)$ cannot exceed the total number of activations. Therefore, Equation (4) accurately computes the maximum number of activations occurring within the interval. □

Listing 1 presents the pseudo-code (python-like) for calculating $\overrightarrow{v^*_{j\to i}}$.

Listing 1. Maximum interference array algorithm.

```
#variables definition and initialization
$\overrightarrow{v^*_{j\to i}}$ = [None]$N_i$
L = [[*range($nT_j$, $nT_j+D_j$,1)] for~n in~range($N_j$)]
if $I_i \neq 0$ and $I_j \neq 0$:
    for~a~in range($N_i$):
        K,g = 0
        if~$aT_i$ in~L:
            K=1
        for~t in~range($aT_i$+1,$aT_i+D_i$, 1):
            if~t % $T_j$ == 0:
                g = g + 1
        $\overrightarrow{v^*_{j\to i}}[a]$ = K + g
```

Once the $\overrightarrow{v^*_{j\to i}}$ vector is defined, it is used to derive the upper bound of the WCRT of task activations. This vector helps us to know whether a receiving task is affected by interference from tasks in other cores. Apart from this vector, we need to know the "delay" that the tasks are subject to due to higher-priority tasks in the same core. For these reasons, we need to identify which activations of the higher-priority tasks overlap with the activations of the task under study.

To do so, we need the following definition and properties.

Definition 3. *Let A_{ij} be a binary matrix of $N_i \times N_j$. The value of $A_{ij_{mn}}$ indicates whether activation m_{th} of τ_i falls within activation n_{th} of τ_j or not in the following way:*

- $A_{ij_{mn}} = 1$*: activation m_{th} of τ_i overlaps with activation n_{th} of τ_j.*
- $A_{ij_{mn}} = 0$*: activation m_{th} of τ_i does not overlap with activation n_{th} of τ_j.*

This matrix serves as the basis to establish when an activation of task τ_i is subject to the influence of activations of tasks with higher priority running in the same core.

Property 1. *If a pair of tasks τ_i and τ_j are allocated to different cores, $M_{\tau_i} \neq M_{\tau_j}$; then, $A_{ij_{mn}} = 0 \; \forall m \in N_i, \forall n \in N_j$.*

Property 2. $A_{ij} = A_{ji}^T \quad \forall \tau_i, \tau_j \in \tau$.

Property 3. $A_{ij} = O$ *if* $i = j$.

We shall demonstrate the behavior of A with an example. We consider task set $\tau'' = [\tau_0'', \tau_1'' \tau_2'']$, with $\tau_0'' = (1, 2, 3, 1)$, $\tau_1'' = (2, 5, 5, 0)$, and $\tau_2'' = (1, 3, 5, 1)$, allocated to a dual-core system, where τ_0'' and τ_1'' are allocated to M_0'', and τ_2'', to M_1'' (Table 3).

Table 3. System with task set τ''.

Task τ'	C	D	T	I	Core M
0	1	2	3	1	0
1	2	5	5	0	0
2	1	3	5	1	1

Figure 4 shows how tasks are allocated to cores and the relation of deadlines and periods. For simplicity, neither computation times nor interferences are depicted. Let us deduce the values of A_{01}, A_{02}, A_{10}, A_{12}, A_{20}, and A_{21}. If tasks are not allocated to the same core, A_{ij} are null matrices. Then, $A_{02} = A_{12} = A_{20} = A_{21} = O$. Let us now calculate A_{10}.

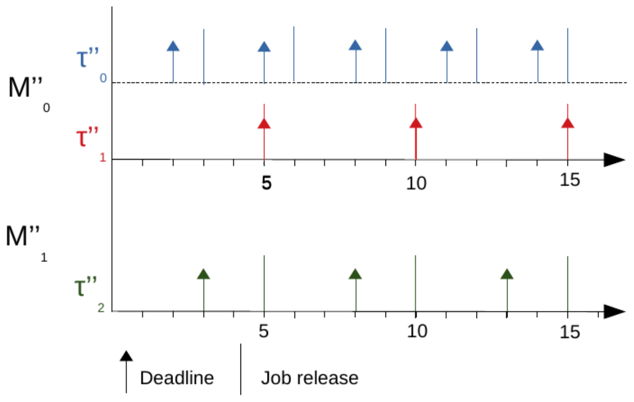

Figure 4. Representation of deadlines and periods of system $\tau'' = [\tau_0'', \tau_1'' \tau_2'']$ allocated to a dual-core platform to derive matrices A_{ij}.

A_{10} is the matrix that relates τ_1 and τ_0 and is composed of N_1 rows and N_0 columns, where $N_1 = H/T_1 = 3$ and $N_0 = H/T_0 = 5$. For each element $a_{10_{ij}}$, it is evaluated if activation i^{th} of τ_1'' overlaps with activation j^{th} of τ_0'', and it may be deduced by observing Figure 4. For example, activation 0 of τ_1'' overlaps with activation 1 of τ_0''. Then, $a_{10_{01}} = 1$. But activation 1 of τ_1'' does not overlap with activation 1 of τ_0''. Then, $a_{10_{11}} = 0$. By evaluating the overlap among all activations of both tasks, A_{10} is obtained.

$$A_{10} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The next section presents the definition of WCRT considering the interference produced in multicore systems, making use of the above definitions.

4.2.2. Worst-Case Response Time with Interference Considerations

In this section, we present a schedulability test based on WCRT by using the definitions of activation pattern array ($\overrightarrow{v^*_{j\to i}}$) and binary matrix ($A_{ij}$).

As stated at the beginning of Section 4.2, the WCRT of an activation of a task depends on its own C_i, the interference from broadcasting tasks allocated to other cores that it is affected by (if it is a receiving task), and the computation times of and interference from the higher-priority tasks allocated to its core. With these three factors, Equation (7) is formulated.

Definition 4. *The upper bound of the WCRT of activation k of task τ_i considering multicore system interference is*

$$WCRT_i^{k^{(ub)}} = C_i + \sum_{\substack{\forall \tau_z \notin M_{\tau_i} \\ I_i \neq 0}} v^*_{z\to i}[k] \cdot I_z + \sum_{\substack{\forall \tau_j \in hp(i) \\ \tau_j \in M_{\tau_i}}} \sum_{a=\left\lfloor \frac{k \cdot T_i}{T_j} \right\rfloor}^{\left\lceil \frac{k \cdot T_i + D_i}{T_j} \right\rceil - 1} \left(C_j + \sum_{\substack{\forall \tau_l \notin M_{\tau_j} \\ I_j \neq 0}} v^*_{l\to j}[a] \cdot I_l \right) \cdot A_{ij_{ka}} \tag{7}$$

As this is an upper bound, it considers that whenever there can be interference, there is interference. In reality, it is possible that even if two tasks on different cores have overlapping execution windows, one of them has already finished its activation when the other one starts its own. That is why the above expression is an upper bound. In addition, it is also important to note that in systems where there is no interference, this definition also gives us an upper bound and it will not give an exact result like Equation (2).

Once interference is considered in the response time to provide an upper bound, the schedulability test presented next applies.

Theorem 2. *A task set τ is schedulable by fixed priorities if*

$$WCRT_i^{ub} = \max_k \left(WCRT_i^{k^{(ub)}} \right) \leq D_i \quad \forall k \in N_i, \forall \tau_i \in \tau \tag{8}$$

Proof. As stated before, the WCRT of a task in multicore systems with interference depends on the following three parameters, which are represented in the expression of $WCRT^{k^{(ub)}}$:

1. The task's computation time: C_i.
2. The interference the task is affected by: $\sum_{\substack{\forall \tau_z \notin M_{\tau_i} \\ I_i \neq 0}} v^*_{z\to i}[k] \cdot I_z$.
3. The computation time of higher-priority tasks allocated to the task's core and the interference they are affected by:
$\sum_{\substack{\forall \tau_j \in hp(i) \\ \tau_j \in M_{\tau_i}}} \sum_{a=\left\lfloor \frac{k \cdot T_i}{T_j} \right\rfloor}^{\left\lceil \frac{k \cdot T_i + D_i}{T_j} \right\rceil - 1} \left(C_j + \sum_{\substack{\forall \tau_l \notin M_{\tau_j} \\ I_j \neq 0}} v^*_{l\to j}[a] \cdot I_l \right) \cdot A_{ij_{ka}}$.

The last two terms are upper bounds on the real interference since they count the maximum number of possible overlaps of execution among tasks. There is no possibility of further interference, as this would imply a higher number of overlaps and the maximum is already calculated in both expressions.

Therefore, since $WCRT_i^{ub}$ is an upper bound to $WRCT_i$ it means that $WCRT_i^{ub} \geq WCRT_i \quad \forall k \in N_i$.

In general, if $WCRT_i \leq D_i \quad \forall \tau_i \in \tau$, then the system is schedulable. So, if $WCRT_i^{ub} \leq D_i$, the system is schedulable, as all the deadlines are met. $\square$

Example

Let us show the usage of Equation (7) to determine the upper bound of the WCRT of all the activations involved in the system previously defined, i.e., τ''. $\tau'' = [\tau_0'', \tau_1''\tau_2'']$, with $\tau_0'' = (1,2,3,1)$, $\tau_1'' = (2,5,5,0)$, and $\tau_2'' = (1,3,5,1)$, allocated to a dual-core system, where τ_0'' and τ_1'' are allocated to M_0'', and τ_2'', to M_1''. Considering that tasks are scheduled according to the DM algorithm, the scheduling plan is shown in Figure 5.

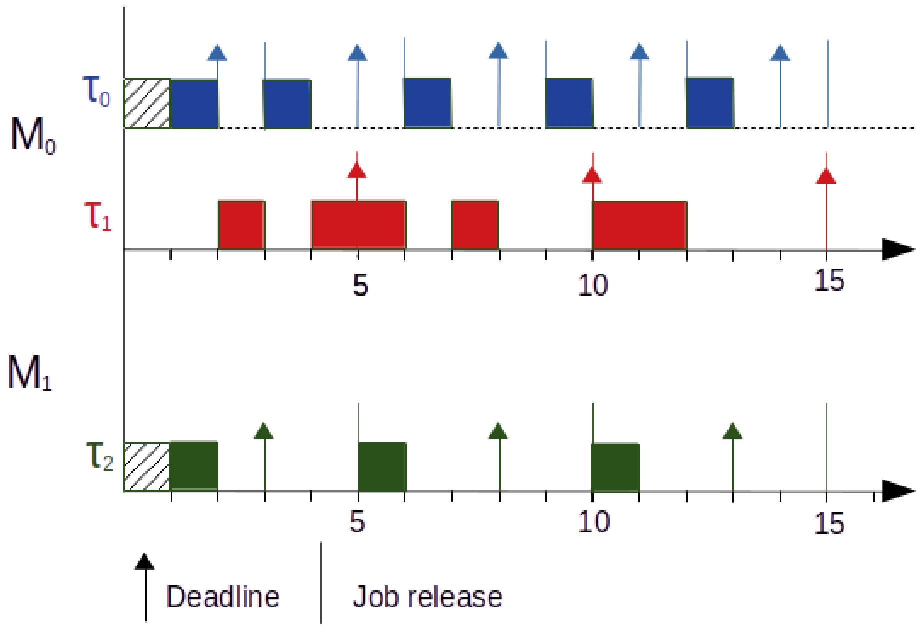

Figure 5. Scheduling plan of system $\tau'' = [\tau_0'', \tau_1''\tau_2'']$ according to DM algorithm.

From the scheduling plan, we can deduce the actual WCRT of all activations of all tasks: $WCRT_0 = (2,1,1,1,1)$, $WCRT_1 = (5,3,2)$, and $WCRT_2 = (2,1,1)$. (Note that the array of WCRT details the WCRT of each activation. For example, $WCRT_1 = (5,4,2)$ means that the WCRT of the first activation of τ_1 is 5, that of the second activation is 4, and that of the third activation is 2.)

In order to calculate the upper bound of WCRT, first, we calculate $\overrightarrow{v_{j\to i}^*}$ and binary matrix A_{ij}. By applying Theorem 1, $\overrightarrow{v_{2\to 0}^*} = (1,0,1,1,1)$ and $\overrightarrow{v_{0\to 2}^*} = (1,1,2)$. τ_1 is not a broadcasting task, so $\overrightarrow{v_{j\to 1}^*} = \overrightarrow{v_{1\to i}^*} = 0$. Array A_{10} was calculated at the end of Section 4.2.1 (we avoid the calculation of other arrays because they do not allow for actions in the formula of WCRT) as

$$A_{10} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

With these above concepts, we can now apply Equation (7) to calculate $WCRT_i^{k^{(ub)}} \ \forall k, i$.

$$WCRT_0^{k^{(ub)}} = C_0 + v_{2\to 0}^*[k] \cdot I_2 + \varnothing$$

$$WCRT_1^{k^{(ub)}} = C_1 + \varnothing + \sum_{a=\left\lfloor \frac{k\cdot 5}{3} \right\rfloor}^{\left\lceil \frac{k\cdot 5+5}{3} \right\rceil - 1} (C_0 + v_{2\to 0}^*[a] \cdot I_2) \cdot A_{10_{ka}}$$

$$WCRT_2^{k^{(ub)}} = C_2 + v_{0\to 2}^*[k] \cdot I_0 + \varnothing$$

Therefore, $WCRT_0^{ub} = (2,1,2,2,2)$, $WCRT_1^{ub} = (5,6,6)$, and $WCRT_2^{ub} = (2,2,3)$.

It can be observed that $WCRT_i^{ub} \geq WCRT_i$ for all tasks and activations. However, τ_1 does not pass schedulability test (2), as $WCRT_1^{1^{(ub)}} = WCRT_1^{2^{(ub)}} = 6 \geq D_1 = 5$, but the system is schedulable. So, the condition of schedulability is sufficient but not necessary.

5. Evaluation

In this section, we evaluate the proposed schedulability test in an experimental environment, first by means of a synthetic load generator and then by means of a real platform.

5.1. Simulations with Synthetic Workload

In this section, we present the evaluation of the proposed schedulability test with a synthetic load in an experimental environment, as depicted in Figure 6. First, the load was generated. Then, for each task set, the scheduling problem was solved. Finally, the results were validated to evaluate some parameters. In parallel, the proposed schedulability test was applied to the task sets, and all results were compared.

Let us explain each of these steps.

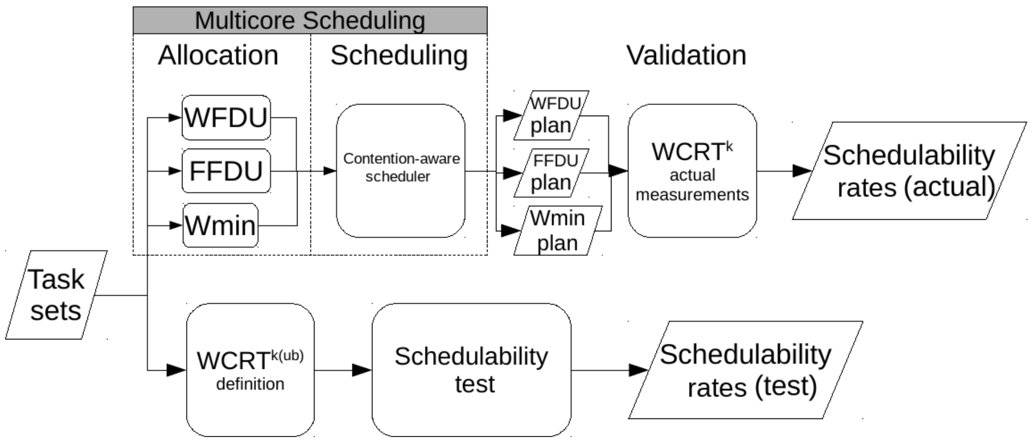

Figure 6. Experimental simulation.

We used the same synthetic task generator as in [9] with the experimental parameters defined in Table 4.

With the total system utilization and a specified number of tasks for each set, the utilization was distributed among the tasks by using the UUniFast discard algorithm from [32]. Periods were randomly generated within the range [20, 1000], and computation times were determined based on system utilization. Deadlines were constrained to be less than or equal to periods, and they were set to $D_i \in [0.5T_i, T_i]$.

The theoretical utilization ranges from 50 and 75% of the system's maximum potential load. For instance, the maximum load of a system with eight cores is 8; so, for evaluation purposes, the initial utilization was set to the range 4.1 (≈50%) to 6 (75%).

Table 4. Definition of experimental scenarios.

Number of Cores	Utilization	Tasks	Broadcasting Tasks	Interference with WCET (%)	Scenario
2	1.1	4	2	10	1
				20	2
				30	3
	1.5			10	4
				20	5
				30	6
4	2.4	12	3	10	7
				20	8
				30	9
	3			10	10
				20	11
				30	12
6	3.1	16	4	10	13
				20	14
				30	15
	4.5			10	16
				20	17
				30	18
8	4.1	20	5	10	19
				20	20
				30	21
	6			10	22
				20	23
				30	24
10	5.1	28	7	10	25
				20	26
				30	27
	7.5			10	28
				20	29
				30	30

The number of broadcasting tasks was configured to be 25% of the total number of tasks, except for scenarios 1–6 (two cores), where it was set to 50%. In the latter case, if only one task is a broadcasting task, no interference is generated. We evaluated every combination of core count and utilization by testing interference levels of 10%, 20%, and 30% relative to the WCET. It is important to note that not all tasks in a task set have the same interference value, but they all experience the same percentage of interference with the WCET.

Once the load was generated, to solve the scheduling problem, tasks were allocated to cores by different algorithms. On the one hand, we considered bin-packing algorithms such as Worst Fit (WF) and First Fit (FF) [7,8] to allocate the tasks to cores. Tasks are ordered in descending order of utilization (DU). On the other hand, the interference-aware Wmin algorithm [4] assigns tasks to cores aiming to reduce contention.

Following the task allocation phase, we now move to the task scheduling phase. For this purpose, we employed the contention-aware scheduling algorithm, as also proposed in [4]. Specifically, we chose Deadline Monotonic (DM) as the priority-based algorithm serving as the basis for this scheduling approach.

Next, the obtained scheduling plans were evaluated, and different parameters were measured (validation phase). In this case, we focused on measuring the response times of the activations of the tasks.

Independently of the above, the schedulability test stated in Theorem 2 was applied to the task sets. This was compared with the results obtained in the validation phase.

Figure 7 shows the actual schedulability rates measured in the validation phase. As expected, when the number of cores and broadcasting tasks increased, the percentage of schedulability decreased. In fact, in systems at higher loading, fixed-priority scheduling techniques cannot guarantee all the deadlines, unlike other algorithms, such as Earliest Deadline First [30,33]. The WFDU allocator is the one that provided better schedulability rates than any other of the proposed algorithms. The FFDU allocator in combination with DM is the algorithm with the worst schedulability rates. The peaks of the function imply scenarios with lower-interference factors, so the schedulability rates are higher.

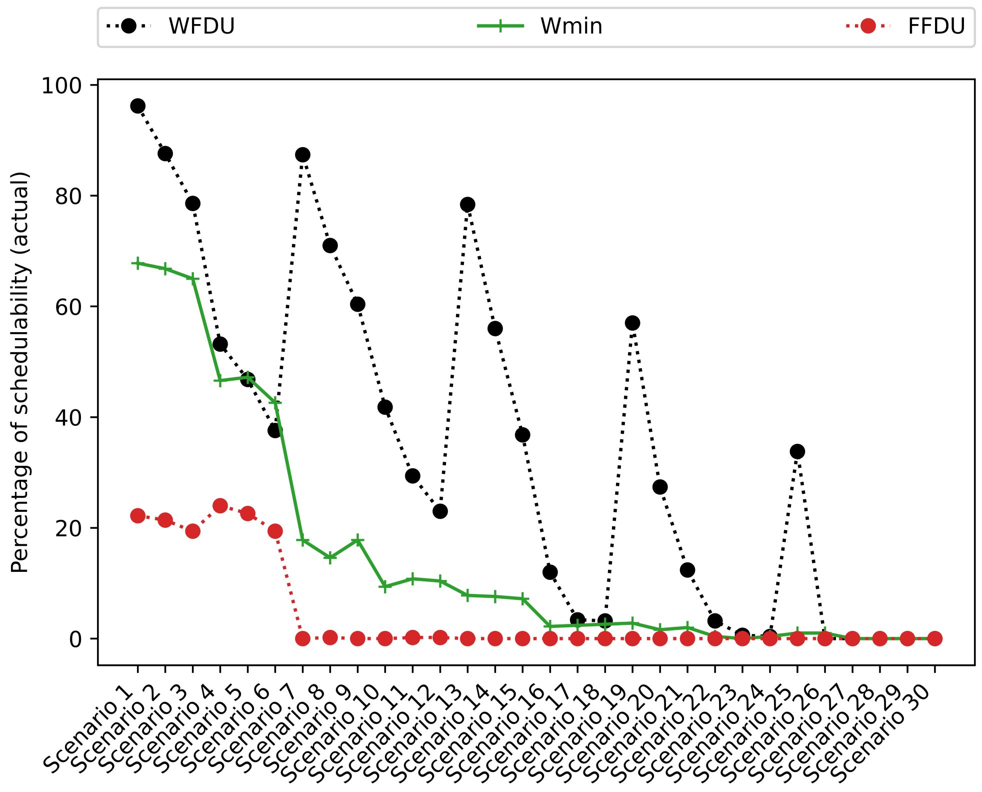

Figure 7. Schedulability ratio with DM and different allocators.

Figure 8 depicts the percentage of tasks that passed the schedulability test over those tasks that were feasibly scheduled. Obviously, if the task set is not schedulable, it makes no sense to apply the schedulability test. As seen in this figure, it is evident that as the number of cores in the system increases, a greater number of tasks fail the schedulability test. This occurs because the definition of $WCRT^{k^{(ub)}}$ is an upper bound, and it is overestimated as the number of broadcasting tasks and the coefficient of interference increase. For FF scenarios, as the schedulability ratio was zero from systems with four cores (from scenario 4 onwards), the schedulability test could not be applied (0 in Figure 8) The WF allocator is the one with which the test provided better results. It is important to note that there is no set that passes the schedulability test and is not schedulable. However, there are task sets that fail the test and are schedulable. Then, the proposed test is sufficient but not necessary.

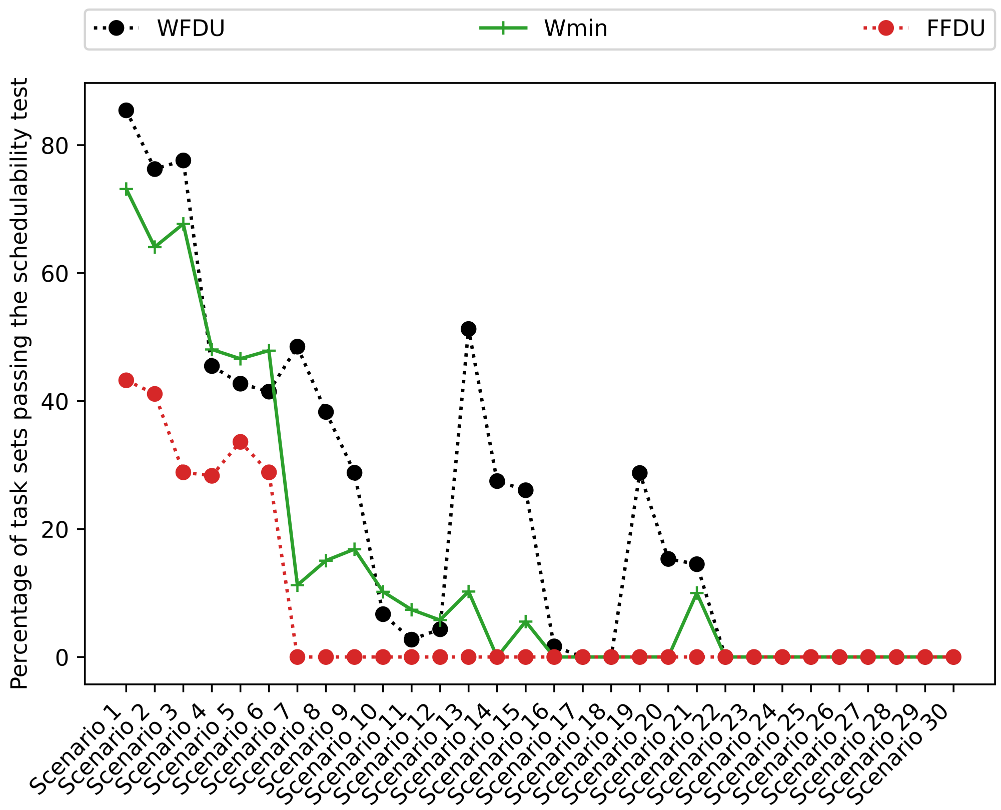

Figure 8. Percentage of cases that pass the proposed schedulability test.

5.2. Simulations On A Real Platform

Besides the simulation environment, a real platform was used to test the proposed contributions. We used a Zybo z7 (Digilent, Tokyo, Japan), which integrates a dual-core ARM Cortex-A9 processor.

We implemented four real-time tasks in Ada language, which is a programming language used for critical software. In particular, we used the Ravenscar profile [34] to restrict the use of many tasking facilities so that the execution of the program was predictable. The parameters of the four tasks are detailed in Table 5 and were created based on a two-core scenario due to the board characteristics. In order to introduce interference, 50% of the tasks were broadcasting tasks.

The code that executed the tasks consisted of loops with mathematical operations read and stored in a data array until the task execution time was completed.

Table 5. Task set parameters (ms) for the experimental use case.

Id	C	D	T	I
0	52	300	300	14
1	11	300	300	0
2	52	400	400	5
3	11	400	400	0

These tasks were allocated to the Zybo dual-core platform by the previously mentioned algorithm, WFDU (Figure 9).

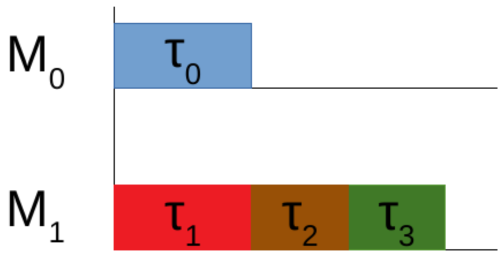

Figure 9. WFDU allocation applied to the task set defined in Table 5.

For this task set, the hyperperiod was $H = 1200$. Then, we calculated the $WCRT$ measured on the platform and the $WCRT^{ub}$ of each k activation. The results are shown in Table 6.

Table 6. $WCRT$ measured on the platform and the $WCRT^{ub}$ of each k activation when the task set is allocated with the WFDU algorithm (Figure 9).

(a) Measured $WCRT$

	Activation k			
	0	**1**	**2**	**3**
τ_0	55.707	59.031	55.704	55.713
τ_1	10.0023	10.00231	10.00231	10.00231
τ_2	60.688	50.4036	50.3931	-
τ_3	81.0023	81.0021	81.0021	-

(b) $WCRT^{ub}$

	Activation k			
	0	**1**	**2**	**3**
τ_0	57	62	62	57
τ_1	11	11	11	11
τ_2	102	102	102	-
τ_3	130	135	130	-

From these results, we can deduce that the proposed upper bound is, in all cases, greater than the actual values measured on the platform. Apart from that, when all parameters in the system are integers, we may assume without loss of generality that all preemptions occur at integer time values. So, throughout this paper, we have assumed that all parameters are indeed integers. However, on a real platform using Ada, we make use of the package Ada.Real_Time, which has a reliable clock for real-time applications.

6. Conclusions

This paper proposes a schedulability assessment utilizing worst-case response time analysis for hard real-time multicore systems. This analysis assumes fixed-priority and constrained-deadline task models. We assume a general model in which interference is a temporal parameter for each task. In other works, schedulability analysis has previously been proposed for this model for dynamic priorities but did not exist for fixed priorities. With this work, we complete the schedulability analysis of the general interference model proposed in [4]. The proposed test is sufficient, as it is an upper bound of the response time of task sets allocated in multicore systems, where tasks suffer from contention with other tasks running simultaneously in other cores. The test was tested with simulations with synthetic workloads and simulations on a real platform.

As future work, we would like to explore how to expand our work into the areas of power consumption and interference reduction. To tackle the consumption problem, frequency reduction in idle instants of time might be a feasible approximation. Also, incorporating the modeling of the consumption of each task in the scheduling phase could lead to interesting results. To lessen the effect of interference, the use of AI techniques might lead to a breakthrough in this aspect of scheduling, mainly based on mathematical models.

Author Contributions: Conceptualization, L.O., A.G., P.B., J.S. and A.C.; methodology, L.O., A.G., P.B., J.S. and A.C.; software, L.O., A.G. and P.B.; validation, L.O., A.G. and P.B.; formal analysis, L.O., A.G., P.B., J.S. and A.C.; investigation, L.O., A.G., P.B., J.S. and A.C.; resources, L.O., A.G., P.B., J.S. and A.C.; data curation, L.O., A.G., P.B., J.S. and A.C.; writing—original draft preparation, L.O., A.G. and P.B.; writing—review and editing, L.O., A.G., P.B., J.S. and A.C.; visualization, L.O., A.G., P.B., J.S. and A.C.; supervision, P.B., J.S. and A.C.; project administration, A.G.; funding acquisition, P.B., J.S. and A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by MCIN/AEI/10.13039/501100011033/ grant PID2021-124502OB-C41 (PRESECREL), and by PAID-10-20 (Universitat Politècnica de València).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request. The data are not publicly available due to privacy.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

WCRT	worst-case response time
WCET	worst-case execution time
DM	Deadline Monotonic
WF	Worst Fit
FF	First Fit
WFDU	Worst-Fit Decreasing Utilization
FFDU	First-Fit Decreasing Utilization

References

1. Dasari, D.; Akesson, B.; Nélis, V.; Awan, M.A.; Petters, S.M. Identifying the sources of unpredictability in COTS-based multicore systems. In Proceedings of the 2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES), Porto, Portugal, 19–21 June 2013; pp. 39–48. [CrossRef]
2. Karuppiah, N. The Impact of Interference due to Resource Contention in Multicore Platform for Safety-critical Avionics Systems. *Int. J. Res. Eng. Appl. Manag. (IJREAM)* **2016**, *2*, 39–48.
3. Kim, H.; de Niz, D.; Andersson, B.; Klein, M.; Mutlu, O.; Rajkumar, R. Bounding memory interference delay in COTS-based multi-core systems. In Proceedings of the 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), St. Louis, MI, USA, 22–24 April 2014; pp. 145–154. [CrossRef]
4. Aceituno, J.M.; Guasque, A.; Balbastre, P.; Simó, J.; Crespo, A. Hardware resources contention-aware scheduling of hard real-time multiprocessor systems. *J. Syst. Archit.* **2021**, *118*, 102223. [CrossRef]
5. Johnson, D.S. Near-Optimal Bin Packing Algorithms. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1973.
6. Aceituno, J.M.; Guasque, A.; Balbastre, P.; Blanes, F.; Pomante, L. Optimized Scheduling of Periodic Hard Real-Time Multicore Systems. *IEEE Access* **2023**, *11*, 30027–30039. [CrossRef]
7. Oh, Y.; Son, S.H. Allocating Fixed-Priority Periodic Tasks on Multiprocessor Systems. *Real-Time Syst.* **1995**, *9*, 207–239. [CrossRef]
8. Coffman, E.G.; Garey, M.R.; Johnson, D.S., Approximation Algorithms for Bin Packing: A Survey. In *Approximation Algorithms for NP-Hard Problems*; PWS Publishing Co.: Boston, MA, USA, 1996; pp. 46–93.
9. Guasque, A.; Aceituno, J.M.; Balbastre, P.; Simó, J.; Crespo, A. Schedulability Analysis of Dynamic Priority Real-Time Systems with Contention. *J. Supercomput.* **2022**, *78*, 14703–14725. [CrossRef]
10. Bril, R.; Lukkien, J.; Verhaegh, W. Worst-Case Response Time Analysis of Real-Time Tasks under Fixed-Priority Scheduling with Deferred Preemption Revisited. *Real-Time Syst.* **2007**, *42*, 269–279.

11. Davis, R.I.; Burns, A. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. *ACM Comput. Surv.* **2011**, *43*, 1–44. [CrossRef]
12. Lugo, T.; Lozano, S.; Fernández, J.; Carretero, J. A Survey of Techniques for Reducing Interference in Real-Time Applications on Multicore Platforms. *IEEE Access* **2022**, *10*, 21853–21882. [CrossRef]
13. Pan, X.; Mueller, F. NUMA-aware memory coloring for multicore real-time systems. *J. Syst. Archit.* **2021**, *118*, 102188. [CrossRef]
14. Hassan, M. Reduced latency DRAM for multi-core safety-critical real-time systems. *Real-Time Syst.* **2019**, *56*, 171–206. [CrossRef]
15. Mancuso, R.; Dudko, R.; Betti, E.; Cesati, M.; Caccamo, M.; Pellizzoni, R. Real-time cache management framework for multi-core architectures. In Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA, USA, 9–11 April 2013; pp. 45–54. [CrossRef]
16. Sun, G.; Shen, J.; Veidenbaum, A.V. Combining prefetch control and cache partitioning to improve multicore performance. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil 20–24 May 2019; pp. 953–962. [CrossRef]
17. Yun, H.; Yao, G.; Pellizzoni, R.; Caccamo, M.; Sha, L. Memory Bandwidth Management for Efficient Performance Isolation in Multi-Core Platforms. *IEEE Trans. Comput.* **2015**, *65*, 562–576. [CrossRef]
18. Xu, M.; Gifford, R.; Phan, L.T.X. Holistic multi-resource allocation for multicore real-time virtualization. In Proceedings of the 56th Annual Design Automation Conference, Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6. [CrossRef]
19. Altmeyer, S.; Davis, R.I.; Indrusiak, L.; Maiza, C.; Nelis, V.; Reineke, J. A generic and compositional framework for multicore response time analysis. In Proceedings of the Proceedings of the 23rd International Conference on Real Time and Networks Systems, Dortmund, Germany, 7–8 June 2015; pp. 129–138. [CrossRef]
20. Negrean, M.; Schliecker, S.; Ernst, R. Response-time analysis of arbitrarily activated tasks in multiprocessor systems with shared resources. In Proceedings of the 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, France, 20–24 April 2009; pp. 524–529. [CrossRef]
21. Schranzhofer, A.; Pellizzoni, R.; Chen, J.J.; Thiele, L.; Caccamo, M. Worst-case response time analysis of resource access models in multi-core systems. In Proceedings of the Design Automation Conference, Anaheim, CA, USA, 13–18 July 2010; pp. 332–337. [CrossRef]
22. Choi, J.; Kang, D.; Ha, S. Conservative modeling of shared resource contention for dependent tasks in partitioned multi-core systems. In Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany 14–18 March 2016; pp. 181–186. [CrossRef]
23. Chen, J.J. Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks. In Proceedings of the 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), Toulouse, France, 5–8 July 2016; pp. 251–261. [CrossRef]
24. Chen, J.; Du, C.; Xie, F.; Yang, Z. Schedulability Analysis of Non-Preemptive Strictly Periodic Tasks in Multi-Core Real-Time Systems. *Real-Time Syst.* **2016**, *52*, 239–271. [CrossRef]
25. Huang, W.H.; Chen, J.J.; Reineke, J. MIRROR: Symmetric timing analysis for real-time tasks on multicore platforms with shared resources. In Proceedings of the 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 5–9 July 2016; pp. 1–6. [CrossRef]
26. Choi, J.; Ha, S. Worst-Case Response Time Analysis of a Synchronous Dataflow Graph in a Multiprocessor System with Real-Time Tasks. *ACM Trans. Des. Autom. Electron. Syst.* **2017**, *22*, 1–26. [CrossRef]
27. Foughali, M.; Hladik, P.E.; Zuepke, A. Compositional verification of embedded real-time systems. *J. Syst. Archit.* **2023**, *142*, 102928. [CrossRef]
28. Andersson, B.; Kim, H.; Niz, D.D.; Klein, M.; Rajkumar, R.R.; Lehoczky, J. Schedulability Analysis of Tasks with Corunner-Dependent Execution Times. *ACM Trans. Embed. Comput. Syst.* **2018**, *17*, 1–29. [CrossRef]
29. Al-bayati, Z.; Sun, Y.; Zeng, H.; Natale, M.D.; Zhu, Q.; Meyer, B.H. Partitioning and Selection of Data Consistency Mechanisms for Multicore Real-Time Systems. *ACM Trans. Embed. Comput. Syst.* **2019**, *18*, 1–28. [CrossRef]
30. Liu, C.L.; Layland, J.W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* **1973**, *20*, 46–61. [CrossRef]
31. Joseph, M.; Pandya, P. Finding Response Times in a Real-Time System. *Comput. J.* **1986**, *29*, 390–395. [CrossRef]
32. Davis, R.I.; Burns, A. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In Proceedings of the 2009 30th IEEE Real-Time Systems Symposium, Washington, DC, USA, 1–4 December 2009; pp. 398–409. [CrossRef]
33. Buttazzo, G.C. Rate Monotonic vs. EDF: Judgment Day. *Real-Time Syst.* **2003**, *29*, 5–26. [CrossRef]
34. Burns, A.; Dobbing, B.; Vardanega, T. Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems. *Ada Lett.* **2004**, *24*, 1–74. [CrossRef]