Fixed-priority global scheduling for mixed-criticality real-time systems

Owen R. Kelly and Hakan Aydin*

Department of Computer Science,
George Mason University,
Fairfax, Virginia 22030, USA
E-mail: okelly@masonlive.gmu.edu
E-mail: aydin@cs.gmu.edu
*Corresponding author

Abstract: There has been a growing interest in recent years in *mixed-criticality* real-time systems in which tasks are attributed different levels of criticality based on the degree to which their deadlines must be assured. While most of the initial efforts on mixed-criticality systems targeted single-processor systems, the research community has recently started to investigate multiprocessor mixed-criticality real-time systems. In this paper we investigate how global, fixed-priority algorithms can be applied in the context of multiprocessor mixed-criticality systems. We identify the two key dimensions of the problem – *priority assignment* and *schedulability testing*, and consider candidate algorithms for each dimension. We also propose a new and simple priority assignment policy called *CPRatio* that considers both task criticality and timing constraints to improve the schedulability of mixed-criticality task sets. We experimentally evaluate the performance of priority assignment strategies and schedulability tests in the global multiprocessor mixed-criticality scheduling context.

Keywords: mixed-criticality; real-time scheduling; global scheduling; fixed-priority scheduling; multiprocessor scheduling; schedulability tests.

Reference to this paper should be made as follows: Kelly, O.R. and Aydin, H. (2014) 'Fixed-priority global scheduling for mixed-criticality real-time systems', *Int. J. Embedded Systems*, Vol. 6, Nos. 2/3, pp.266–276.

Biographical notes: Owen R. Kelly received his Bachelor of Engineering degree in Electrical Engineering from the State University of New York at Stony Brook in 1983, and his Master of Science degree in Electrical Engineering from Polytechnic University in New York in 1989, and his Master of Science degree in Computer Science from George Mason University in 2009. Over the past 30 years, he has worked on the design and implementation of software systems in a variety of application areas.

Hakan Aydin is an Associate Professor of Computer Science at George Mason University. He received his PhD in Computer Science from the University of Pittsburgh. He has published more than 60 peer-reviewed papers and served on the programme committees of several international conferences. He received the NSF CAREER Award in 2006. He was the Technical Programme Chair of IEEE RTAS in 2011. His current research focuses on low-power computing, real-time systems and fault tolerance.

1 Introduction

Hard real-time systems – systems for which the completion of tasks by specified deadlines is fundamental to correct operation, have been a focus of intensive research for several decades. Real-time applications are typically periodic in nature, comprised of recurrent tasks that each generate a sequence of instances or jobs. With *fixed-priority* (FP) scheduling algorithms, all jobs of a task are assigned the same priority relative to jobs of other tasks, while in *dynamic-priority* approaches, the priority of a task can vary over time. The dynamic-priority algorithm *earliest-deadline-first* (EDF) is known to be optimal for the uniprocessor scheduling of sets of periodic, independent tasks with deadlines equal to periods (i.e., implicit deadlines) in the sense that any task set that is schedulable by *some* algorithm is also schedulable by EDF (Liu and Layland, 1973). Among FP algorithms, the rate-monotonic (RM) algorithm is optimal for periodic task sets with implicit deadlines (Liu and Layland, 1973). In RM scheduling, tasks with smaller periods have higher priorities.

Following the recent advent of multi-core systems, the multiprocessor scheduling of real-time applications has received increasing attention (Jersak et al., 2005). Multiprocessor scheduling algorithms are generally classified as *partitioned* or *global* scheduling algorithms. In partitioned scheduling approaches, each task is allocated to

a single processor, and the scheduling problem becomes one of task allocation followed by the uniprocessor scheduling of the subset of tasks allocated to each processor (Lopez et al., 2003, 2004). In fact, an important advantage of partitioned approaches is the applicability of the rich body of results for uniprocessor scheduling after the partitioning phase. However, partitioning of real-time tasks is, in general, an NP-Hard problem. Also, the inability to move tasks to other processors may be problematic if some tasks may arrive or depart dynamically. In contrast, in global scheduling approaches, jobs are scheduled from a single ready queue, and each job of a task may be preempted and later resumed, potentially on a different processor. Global approaches are more suitable for dynamic workloads in which tasks are added and removed over time. On the other hand, the run-time overhead of task migration is a factor. Global and partitioned approaches are *incomparable* in that there are task sets that are schedulable by a global approach but not a partitioned approach, and others for which the opposite is true (Leung and Whitehead, 1982). In addition, many results for uniprocessor scheduling do not extend to multiprocessor systems. For example, Dhall and Liu (1978) observed that there are task sets with total utilisation slightly larger than one that cannot be scheduled by global EDF or RM on a multiprocessor regardless of the number of processors – a phenomenon that has become known as the *Dhall effect*.

Response-time analysis (RTA) was developed as a technique for FP uniprocessor schedulability analysis in which the worst-case response time of each task is calculated assuming a particular priority-assignment scheme and compared with its deadline to determine its schedulability (Joseph and Pandya, 1986). For multiprocessor scheduling, the worst-case response time does not necessarily occur when all jobs arrive at the same time (Lauzac et al., 1998). This fact adds another layer of complexity to global FP scheduling algorithms.

Baker (2003) followed an approach for multiprocessor schedulability testing in which the minimum interference necessary for a task to miss its deadline is determined. An upper bound on the interference experienced by a task is calculated, and if this bound is less than the minimum required for a deadline miss, the task is guaranteed to meet its deadlines. Other researchers have followed and refined the approach established by Baker (e.g., Bertogna and Cirinei, 2007; Bertogna et al., 2009; Guan et al., 2009).

A large majority of the research in real-time systems has been based on an assumption that tasks have the same degree of importance or criticality. Moreover, a single worst-case execution time (WCET) is assigned to each task and deadline guarantees resulting from schedulability analysis are valid provided that no task exceeds its WCET. However, a more recent trend in design involves approaches in which multiple real-time functions at varying *criticality* levels are implemented on the same embedded system platform. This has led to the investigation of the so-called *mixed-criticality* systems that are based on models that explicitly take criticality into account. As an example, in the domain of unmanned aerial vehicles (UAVs), the main functions are divided into two categories: Level 1 functions that are classified as *mission-critical*, and Level 2 functions that are considered to be *flight-critical* (Baruah et al., 2012). In addition, there is often a need to apply rigorous techniques to *certify* these functions in their own criticality range, which is challenging, given that they are implemented on the same embedded hardware platform (and thus, may interact with each other) (Baruah et al., 2012). As such, mixed-criticality systems facilitate the integration of functions with different criticality levels on the same platform (Baruah et al., 2010).

In his seminal work, Vestal (2007) observed that the WCET estimate assigned to a task tends to increase with an increase in the degree to which the task's deadlines must be assured. He proposed a model, further explored by Baruah and Vestal (2008), in which each task is assigned multiple WCETs, each associated with a different level of assurance or criticality. Each task is also assigned a criticality level and the schedulability of a task is evaluated using the WCETs associated with its criticality level. Deadline guarantees are made for low-criticality tasks under the assumption that higher-criticality tasks do not exceed their low-criticality WCETs. That is, low criticality tasks make use of processor capacity that is allocated to higher-criticality tasks but not used by those tasks. This results in an increase in processor utilisation. Several aspects of mixed-criticality uniprocessor systems have been explored by Baruah et al. (2010), Li and Baruah (2010), de Niz et al. (2009) and Lakshmanan et al. (2011).

Research efforts for multiprocessor mixed-criticality systems are relatively new. In our previous work, we investigated the partitioned scheduling of mixed-criticality task sets on multiprocessors using FP algorithms (Kelly et al., 2011). We identified algorithms that result from choices made in the two key dimensions of the problem: task allocation and priority assignment. We established that two heuristics for ordering tasks prior to allocation – by decreasing utilisation or by decreasing criticality, are incommensurable in that there are task sets that are schedulable by one, but not the other. Our results also suggested that Audsley's (1991) algorithm offers significantly improved performance over RM priority assignment in this setting. Other multiprocessor mixed-criticality research efforts include those by Lakshmanan et al. (2010), Mollison et al. (2010) and Pathan (2012).

1.1 This work

Our focus in this paper is the *global* FP *scheduling* of mixed-criticality task sets on homogeneous multiprocessors. We explore the two key dimensions of this problem, namely *priority assignment* to individual tasks and *feasibility testing* given a specific priority assignment. We identify algorithms that can be used in each dimension. We undertake a detailed experimental evaluation of the relative performance of algorithms developed for traditional task sets in each of these dimensions when applied to mixed-criticality task sets

using the model proposed by Vestal (2007). Another contribution of this paper is the proposal of a new priority assignment approach referred to as *criticality-to-period ratio* (*CPRatio*). *CPRatio* targets mixed-criticality scheduling by assigning priorities based on both a task's period and criticality level. We show that the sophisticated priority assignment algorithm developed by Audsley (1991) for uniprocessor scheduling provides significantly better performance relative to other approaches for the global FP scheduling of mixed-criticality tasks when the schedulability test of Bertogna et al. (2009) is applied. On the other hand, *CPRatio* offers improved performance over other approaches while not incurring the additional implementation complexity of Audsley's algorithm. Moreover, when combined with the test proposed by Bertogna and Cirinei (2007), we show that *CPRatio* outperforms all other priority-assignment schemes, including Audsley's scheme.

The rest of this paper is organised as follows. We summarise related work in Section 2. In Section 3, we describe our system model and notation. In Section 4, we provide an example to illustrate the key principles associated with the global FP scheduling of mixed-criticality task sets. In Section 5, we describe the algorithms that have been developed for the assignment of fixed priorities for global scheduling and for schedulability tests. We present the results of our experimental evaluation in Section 6, and we summarise our conclusions in Section 7.

2 Related work

The partitioned scheduling of traditional task sets has been explored in several works, including those by Lopez et al. (2003, 2004). A majority of early efforts in the multiprocessor scheduling of traditional task sets were in fact focused on partitioned scheduling. A number of research efforts in global scheduling have been motivated by a desire to mitigate the Dhall effect, e.g., the works by Andersson and Jonsson (2000, 2003). Baker (2003) derived schedulability conditions for the EDF and deadline monotonic (DM) scheduling of task sets with deadlines less than or equal to periods by considering the conditions that must exist for a job to miss its deadline. This seminal work by Baker was the basis for the subsequent development of several schedulability tests for traditional task sets. Bertogna and Cirinei (2007) presented an approach in which an upper bound on the response-time of a task is derived and compared with the task's relative deadline to determine its schedulability. Guan et al. (2009) proposed a technique that improves upon the approach specified by Bertogna and Cirinei (2007). Bertogna et al. (2009) proposed an approach in which an upper bound on the interference experienced by a task is determined and used to evaluate its schedulability.

Davis and Burns (2011) have shown that the priority assignment algorithm developed by Audsley for uniprocessor scheduling (Audsley, 1991) can be applied to multiprocessor global scheduling if three conditions (which they derive) are met by the schedulability test to be applied. A test that meets these conditions is said to be *compatible with Audsley's algorithm*.

Mixed-criticality research was pioneered by Vestal (2007) and Baruah and Vestal (2008). A load-based approach to the schedulability analysis of uniprocessor mixed-criticality systems was derived by Li and Baruah (2010). A different approach to uniprocessor mixed-criticality scheduling, proposed by de Niz et al. (2009) and further explored by Lakshmanan et al. (2011), involves the off-line calculation of zero-slack instants and an on-line component that effectively adjusts priorities such that higher-criticality tasks receive high priority when zero-slack situations arise.

Lakshmanan et al. (2010) applied mixed-criticality concepts to the handling of workload spikes in distributed cyber-physical systems. They developed a metric for the evaluation of scheduling algorithms referred to as *ductility* and proposed an algorithm named *compress-on-overload packing* that maximises this metric. Mollison et al. (2010) proposed an approach targeted at scheduling on multicore platforms in which the slack resulting from the difference between predicted WCETs and actual execution times is reclaimed for lower-criticality tasks. They developed an architecture for multiprocessor mixed-criticality scheduling that provides temporal isolation between tasks of different criticality levels. Pathan (2012) derived a schedulability test for mixed-criticality, multiprocessor scheduling based on RTA that can be combined with Audsley's (1991) algorithm for the assignment of fixed-priorities. This work differs from the approach we consider in this paper in that it assumes the existence of a run-time mechanism that terminates tasks of a particular criticality level if the criticality level attributed to the behaviour of the system exceeds that level. In Kelly et al. (2011), we explored the partitioned scheduling of mixed-criticality systems, considering the relative importance of the two key dimensions of this problem, namely *task allocation* and *priority assignment*.

3 System model

We consider real-time workloads that are comprised of a set τ of n mixed-criticality tasks τ_1, ..., τ_n. Each task τ_i is comprised of a recurring series of instances or jobs. Tasks are *sporadic*, i.e., consecutive jobs in each task are separated by a minimum inter-arrival time referred to as the task's period. Tasks are assumed to be independent, aside from their execution on shared processors. Each task τ_i has a relative deadline D_i that is equal to its period (T_i); in other words, we consider tasks with implicit deadlines.

We apply the mixed-criticality schedulability model proposed by Vestal (2007) and further explored by Baruah and Vestal (2008). In particular, a task set is comprised of tasks of varying levels of importance or criticality. Each task τ_i is assigned a criticality level L_i that is based on the degree to which the meeting of its deadlines must be

assured. Criticality levels are represented as integers, with larger values denoting higher criticality and with the number of criticality levels denoted as k, with $k \leq n$. Each task is assigned a WCET function C_i that specifies a WCET estimate $C_i(X)$ for each criticality level X in the system, where $C_i(X-1)$ is assumed to be less than or equal to $C_i(X)$ (Vestal, 2007; Baruah and Vestal, 2008). Each task is also associated with a utilisation function u_i, with the utilisation value for criticality level X denoted as $u_i(X)$ and defined as $C_i(X)/T_i$. The sum of the task utilisations at the highest criticality level is referred to as $U_{max}^{tot} = \sum_{i=1}^{n} u_i(k)$.

Tasks are scheduled on a set of m homogeneous processors designated P_i, ..., P_m using global scheduling. In global scheduling, all ready jobs are scheduled from a single priority-ordered queue. If there are more than m ready jobs, then the m highest-priority jobs are each allocated to a distinct processor. An arriving job is placed in the ready queue in priority order, and if it is among the m highest priority jobs, then the lowest-priority executing job is preempted and its processor is allocated to the new job. The preempted job may be later resumed on a potentially different processor. Intra-task parallelism is not permitted, i.e., a task may execute on at most one processor at any time. The response time R_i of a task τ_i is the maximum time from the release of a job of the task to its completion. Tasks are assigned fixed priorities, i.e., each job in a task has the same priority relative to jobs in other tasks. Following the approach of Bertogna and Cirinei (2007) and Bertogna et al. (2009), time is represented by non-negative integers.

4 Motivational example

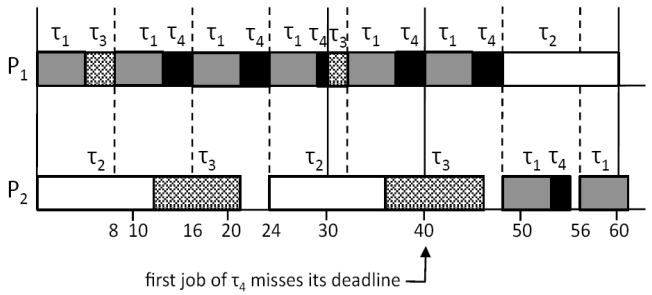
To illustrate the use of the mixed-criticality model proposed by Vestal (2007) in global FP scheduling and the importance of priority assignment in this context, we consider the example four-task set with four criticality levels shown in Table 1.

Table 1 Example mixed-criticality task set

Task	L_i	T_i	$C_i(1)$	$C_i(2)$	$C_i(3)$	$C_i(4)$
τ_1	2	8	3	3	5	5
τ_2	1	24	3	3	12	12
τ_3	4	30	8	8	12	12
τ_4	3	40	6	6	15	15

We first determine whether the example task set is schedulable with the RM-priority ordering τ_1, τ_2, τ_3, τ_4, which is the order in which the tasks are listed in Table 1. Because the mixed-criticality model requires the schedulability of each task to be evaluated using WCETs corresponding to its criticality levels, the schedulability of each task must be evaluated as a separate case. Figure 1 depicts the evaluation of the schedulability of a specific task, τ_4, when RM priority assignment is used, assuming synchronous arrival. Because τ_4 has criticality level 3, criticality-level 3 WCETs are used for all tasks. Specifically, the WCETs of τ_1, τ_2, τ_3, and τ_4 are 5, 12, 12, and 15, respectively. As shown in Figure 1, τ_4 misses its deadline at time 40 under these circumstances and therefore the task set is not schedulable by global RM.

Figure 1 Schedulability analysis with RM priority assignment



In the case above, the WCETs used in the evaluation of τ_4 correspond to the criticality level 3 and are therefore relatively large. If priorities were instead assigned such that the lowest-priority task had a low criticality, the WCETs attributed to higher-priority tasks would tend to be smaller, thereby increasing the likelihood that the lowest-priority task will be deemed schedulable. This suggests that the assignment of priorities based on criticality level might lead to improved schedulability. We consider next whether the example task set is schedulable with priorities assigned in this manner, i.e., with higher-criticality tasks assigned higher priority – a strategy referred to as *criticality monotonic* by de Niz et al. (2009). The resulting task order is τ_3, τ_4, τ_1, τ_2. The evaluation of the schedulability of τ_i is depicted in Figure 2. Because τ_1 has criticality level 2, we use criticality level 2 WCETs for all tasks, i.e., τ_3, τ_4, τ_1, τ_2 are assigned WCETs 8, 6, 3, 3, respectively. Despite the use of these smaller WCETs, as shown in Figure 2, the task set is not schedulable with priorities assigned in accordance with criticality, mainly because a task with a stricter timing constraint (smaller relative deadline), τ_1, is subject to the interference of higher-criticality tasks which happen to have larger periods.

Figure 2 Schedulability analysis with priorities based on criticality

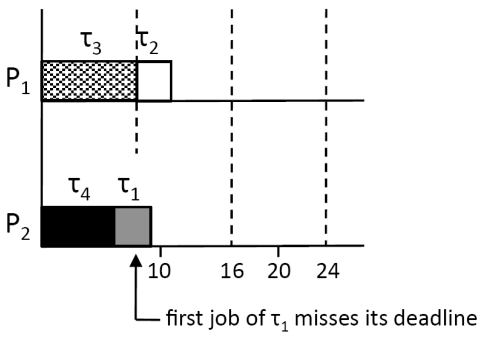
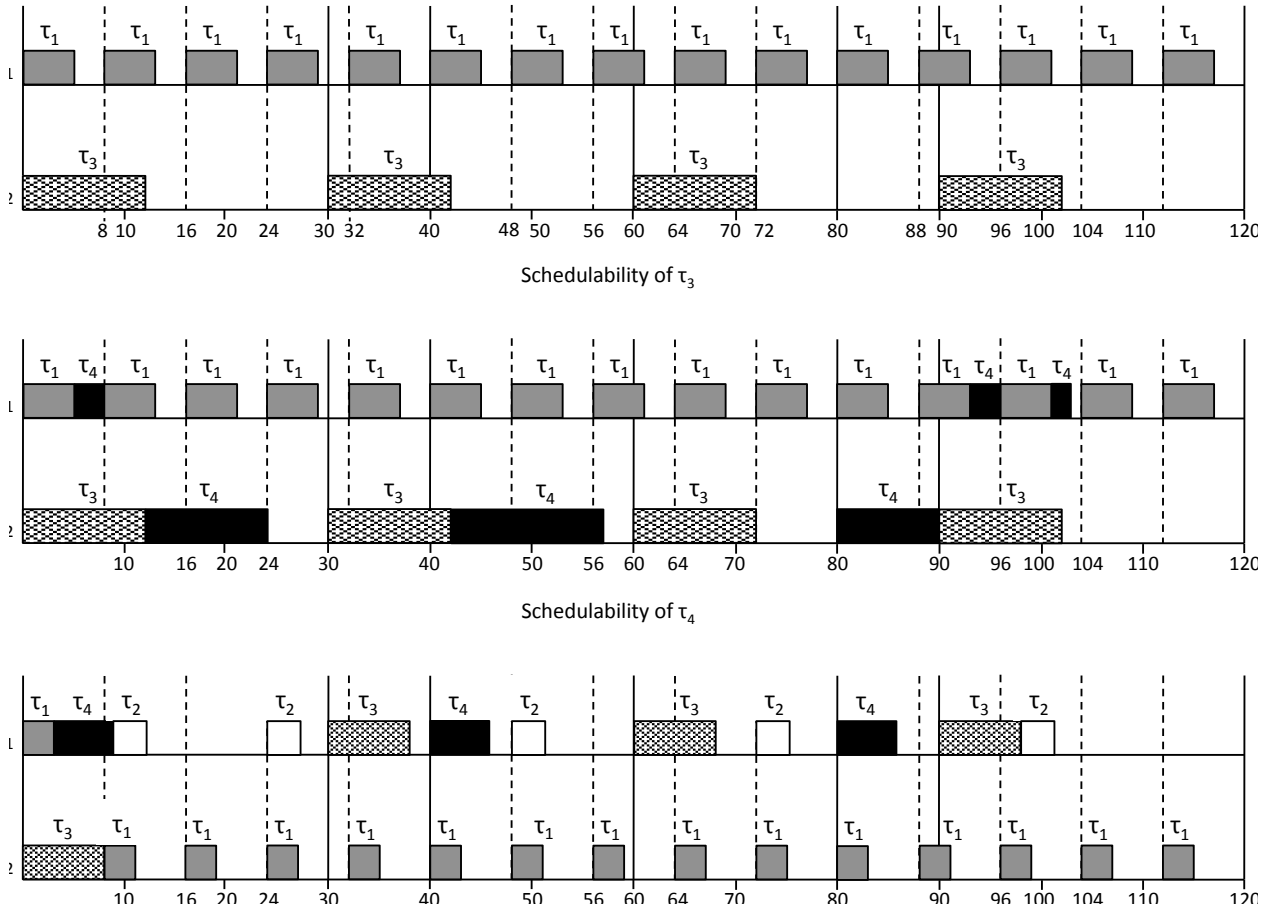


Figure 3 Schedulability analysis with an alternative priority assignment scheme



Finally, we consider an alternative priority ordering: τ_1, τ_3, τ_4, τ_2. The highest priority task, τ_1, meets its deadlines. Figure 3 shows the evaluation with this priority ordering of the schedulability of tasks τ_3, τ_4, and τ_2 in separate subfigures. In each case, the deadlines of all tasks are met. We note that because the depicted synchronous arrival of tasks is not necessarily the worst-case arrival pattern, the fact that no task misses a deadline in the situation considered in Figure 3 does not guarantee that the task set is schedulable. However, we show in Section 5.2, through the application of a sufficient schedulability test that this example task set is indeed schedulable with this alternative assignment of priorities even without the assumption of synchronous arrival and without the strictly-periodic behaviour depicted in the figure. We also note that this specific priority assignment is not entirely arbitrary, but is obtained by considering the ratio of a task's criticality to its period, a scheme that we propose for mixed-criticality task sets and refer to as *CPRatio*.

5 Dimensions of global mixed-criticality FP scheduling

Given that a global, FP approach is to be employed for the multiprocessor scheduling of real-time, mixed-criticality task sets, there are two key dimensions in which an algorithm must be chosen. Specifically, a *priority-assignment strategy* and a *feasibility* (*schedulability*) *test* must be selected. In this section, we consider algorithms that have been proposed in each of these dimensions. We describe the algorithms that we have chosen to evaluate based on their suitability for the scheduling of mixed-criticality task sets as well as our objective of selecting a subset of algorithms that is representative of the research that has been performed in these two dimensions. We also propose a new priority assignment scheme designed specifically for mixed-criticality scheduling.

5.1 Priority assignment

A fundamental principle in evaluating the schedulability of a task in the Vestal (2007) mixed-criticality model is that the WCET estimates used in the evaluation must be those corresponding to the criticality of the task under consideration. That is, when determining the impact of a task τ_i on the schedulability of a lower-priority task τ_k, the WCET of τ_i corresponding to the criticality level of τ_k must be used. Because the WCETs attributed to a task are a non-decreasing function of criticality level, the higher the criticality level of τ_k, the larger will be the potential impact of τ_i on its schedulability. The WCET of τ_i corresponding to the highest criticality level among lower-priority tasks

represents an upper bound on the interference that a job of τ_i can present to lower-priority tasks. For this reason, priority assignment for mixed-criticality scheduling has even greater significance than it does for traditional task sets.

In the following, we describe a representative subset of the priority assignment strategies that have been applied to global, FP scheduling. Some were developed for uniprocessor scheduling and adapted for multiprocessor scheduling, while others were specifically targeted at multiprocessor scheduling. Almost all were developed for traditional task sets. The exceptions are the strategy in which priorities are assigned solely based on criticality, referred to as *criticality monotonic* (CM) priority assignment, and a new priority assignment that we propose, referred to as *CPRatio*.

5.1.1 Rate-monotonic

The RM priority assignment (Liu and Layland, 1973) is widely used in many real-time embedded applications. It is well-known that the optimality of RM on single-processor systems does not extend to multiprocessors (Davis and Burns, 2011). Also, Vestal (2007) showed that RM is not optimal for the FP scheduling of mixed-criticality task sets on a uniprocessor. Despite these limitations, we nevertheless evaluate the application of RM to the global FP scheduling of mixed-criticality task sets.

5.1.2 Criticality-monotonic

An alternative proposed by Vestal (2007) is to assign priorities based on criticalities: the higher the criticality, the higher the scheduling priority. However, as discussed by de Niz et al. (2009), there are task sets that are not schedulable with this priority assignment that are otherwise schedulable by the simple RM algorithm.

5.1.3 Criticality-to-period ratio

The discussion above regarding the potential benefit of assigning priorities in accordance with criticality levels highlights the tension between *period* (a timing parameter) and *criticality level* in priority assignment. In this paper, we propose a priority-assignment metric that *combines the two attributes*. We observe that from a timing point of view, assigning high priority to tasks with smaller periods is, in general, advantageous. Similarly, the interference on high-criticality tasks is in general reduced if they are assigned high priorities. Because both small periods and high criticality levels suggest the assignment of a higher priority, we propose that priorities be assigned to mixed criticality tasks based on the ratio of criticality level to period, with larger values for this metric resulting in the assignment of higher priorities – a strategy that we refer to as *CPRatio*.

In the example presented in Section 4, the task set was not schedulable with RM or CM priority assignment. In this example, the ratio of criticality to period, L_i/T_i, for tasks τ_1, τ_3, τ_4, and τ_2 is 0.25, 0.13, 0.075, and 0.042, respectively, and this is therefore the priority ordering that results from *CPRatio*. When the schedulability tests described in Section 5.2 are applied to this task set with this priority ordering, it is shown to be schedulable. We underline that *CPRatio* is not optimal (there may be task sets that are schedulable by another priority assignment but not with *CPRatio*). On the other hand, as we show through experiments, *CPRatio* offers a more robust performance compared to other schemes, under different feasibility tests.

5.1.4 Audsley's algorithm

Audsley (1991) developed an algorithm for the assignment of fixed priorities in uniprocessor scheduling that is optimal for traditional task sets when tasks do not arrive at the same time. Audsley's algorithm requires that $O(n^2)$ tests be performed for n tasks. For uniprocessor scheduling, each of these feasibility tests may be performed using time demand analysis, which may take pseudo-polynomial time (Liu, 2000).

For mixed-criticality task sets, while RM priority assignment is not optimal, Audsley's algorithm provides quick iteration over possible priority assignments without the loss of optimality on uniprocessor systems (Vestal, 2007). In Kelly et al. (2011), we explored the use of Audsley's algorithm in the partitioned scheduling of mixed-criticality task sets where it was applied following the allocation step for the scheduling of the subset of tasks allocated to each processor. We found that it offered a significant performance gain over RM priority assignment in this setting. Audsley's algorithm is not always optimal for global FP scheduling (Davis and Burns, 2011). Davis and Burns (2011) described the issue of the compatibility of global FP schedulability tests with Audsley's algorithm, which is summarised in Section 5.2 and is important for the understanding of our experimental results.

5.1.5 TkC

The TkC algorithm was defined by Andersson and Jonsson (2000) for global FP scheduling and was motivated by a desire to overcome the Dhall effect. The value of $T_i - kC_i$ is calculated for each task τ_i, where T_i is the task's period, C_i is its WCET, and k is a real number calculated based on the number of processors. The task with the smaller value of $T - kC$ is assigned higher priority. When applied to mixed-criticality task sets, because each task is associated with multiple WCETs, a choice of the WCET to be attributed to each task for the purposes of priority assignment must be made. For example, the WCET of each task corresponding to the highest criticality level in the system could be chosen, in which case we refer to the priority-assignment approach as TkCMax.

5.1.6 D-C monotonic

In the D-C monotonic (DCM) algorithm described by Davis and Burns (2011), priorities are assigned based on the value of $D_i - C_i$ for each task τ_i, with the task with a smaller value for this quantity receiving higher priority. Like TkC, a

WCET must be selected for each task in order to calculate this metric. For example, the WCET associated with the highest criticality level in the system could be used, a scheme that we refer to as DCMMax.

5.2 Schedulability tests

Several research efforts have resulted in the derivation of utilisation bounds that can serve as schedulability tests for the global FP scheduling of traditional task sets. However, the concept of the total utilisation of a task set is less meaningful for mixed-criticality task sets (Baruah and Vestal, 2008). We therefore focus on tests that determine schedulability based on the interference experienced by a task rather than total utilisation. Such tests can be tailored for application to mixed-criticality workloads by calculating interference in accordance with the mixed-criticality model proposed by Vestal (2007).

Baker (2003) developed schedulability tests for both the RM and EDF global scheduling of traditional, sporadic task sets with constrained deadlines by considering the conditions that are necessary for a job to miss its deadline. A lower bound on the load necessary in the interval from the release of a job to its deadline (referred to as the *problem window*) for a deadline miss to occur is calculated.

5.2.1 Bertogna-Cirinei 2007 (BC2007)

The seminal work by Baker (2003) was the basis for the subsequent development of several schedulability tests for global, FP scheduling. Bertogna and Cirinei (2007) derived an approach that is applicable for any priority-assignment scheme and used this approach to specify tests for both EDF and FP strategies. Specifically, a form of RTA is used in which an upper bound on the response time of a task τ_k is calculated by determining an upper bound on the interference presented by other tasks in the problem window of τ_k. As specified in Theorem 7 by Bertogna and Cirinei (2007), an upper bound R_k^{ub} on the response time of a task τ_k is determined by the iterative, fixed-point evaluation of the following expression, beginning with

$$R_k^{ub} = C_k :$$
$$R_k^{ub} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{i<k} \hat{I}_k^i \left(R_k^{ub} \right) \right\rfloor \quad (1)$$

where $\hat{I}_k^i(R_k^{ub})$ is an upper bound on the interference presented by task τ_i on task τ_k in the interval from the release of a job of τ_k to its response time R_k^{ub} and is calculated as

$$\hat{I}_k^i \left(R_k^{ub} \right) = \min \left(W_i \left(R_k^{ub} \right), R_k^{ub} - C_k + 1 \right). \quad (2)$$

Tasks are assumed to be indexed in priority order, so the sum in equation (1) is over tasks with priority higher than that of τ_k. In equation (2), $W_i(R_k^{ub})$ is an upper bound on the workload of task τ_i in an interval of length R_k^{ub}, calculated by assuming that the carried-in job of τ_i completes at its worst-case response time. We refer to the test specified by Bertogna and Cirinei (2007) as BC2007. Guan et al. (2009) proposed an improvement to the RTA test described by Bertogna and Cirinei (2007) in which the number of tasks for which a carried-in job contributes to the interference is limited to $m - 1$. It was shown by Davis and Burns (2011) that the resulting test is not compatible with Audsley's algorithm.

5.2.2 Bertogna et al. 2009 (B2009)

Bertogna et al. (2009) developed another schedulability test by assuming that a carried-in job of τ_i completes at its relative deadline D_i (which we assume, in this paper, is equal to its period T_i) when calculating an upper bound on the workload of τ_i in a problem window. Because it assumes that a carried-in job completes at its response-time upper bound rather than at its deadline, the test developed by Bertogna and Cirinei (2007) provides a tighter upper bound for the workload of a task τ_i in a problem window. However, unlike the test developed by Bertogna and Cirinei (2007), the test proposed by Bertogna et al. (2009) does not require an iterative, fixed-point calculation. Theorem 8 from Bertogna et al. (2009) states that a task set is schedulable with global, FP scheduling on a multiprocessor if each task τ_k in the set satisfies the following schedulability condition:

$$\sum_{i<k} \min \left(W_i \left(T_k \right), T_k - C_k + 1 \right) < m \left(T_k - C_k + 1 \right) \quad (3)$$

where tasks are assumed to be indexed in accordance with priority such that the sum is over tasks with higher priority than τ_k. $W_i(T_k)$ is an upper bound on the workload of τ_i in the problem window. By applying this equation to the example task set presented in Section 4, it can be shown that the task set is schedulable with priorities assigned by *CPRatio*.

5.2.2.1 Compatibility with Audsley's algorithm

One key assumption underlying Audsley's algorithm is that the schedulability of a task can be determined given the subset of tasks that have higher priority, but without knowing the relative priority ordering of those higher-priority tasks. When applied to uniprocessor scheduling, this assumption is valid, but for multiprocessor scheduling, its optimality depends on the schedulability test in conjunction with which it is used. Davis and Burns (2011) derived three conditions that are necessary and sufficient for a schedulability test to be used with Audsley's algorithm. We note that the tests that meet these conditions are referred to as *OPA compatible* in Davis and Burns (2011). For a test to be compatible, its evaluation of the schedulability of a task must not depend on the relative priority ordering of higher priority tasks or on the relative priority ordering of lower-priority tasks. Davis and Burns (2011) have shown that the schedulability test that we refer to as B2009 is

compatible with Audsley's algorithm. B2009 has a time complexity of $O(n^2)$, with a complexity of $O(n)$ for each task. Because the number of tests executed by Audsley's algorithm is $O(n^2)$, and each evaluates the schedulability of a single task, when B2009 is combined with Audsley's algorithm, the resulting, overall complexity is $O(n^3)$ (polynomial-time). It was shown by Davis and Burns (2011) that the test that we refer to as BC2007 is not compatible with Audsley's algorithm. Given that the tighter workload upper bound (relative to B2009) of BC2007 is not applicable when Audsley's algorithm is used for priority assignment, it is important to consider the performance of BC2007 combined with other priority-assignment schemes relative to the performance of B2009 combined with Audsley's algorithm. Specifically, a key question is the following: *does the increased performance expected with Audsley's algorithm compensate for the more pessimistic bound of B2009*? Our experimental results, presented next, address this question.

6 Experimental results

We performed an experimental evaluation of the performance of the priority assignment algorithms and schedulability tests described in Section 5 when applied to the global multiprocessor scheduling of mixed-criticality task sets. In this section, we describe our methodology and present the results of our investigation.

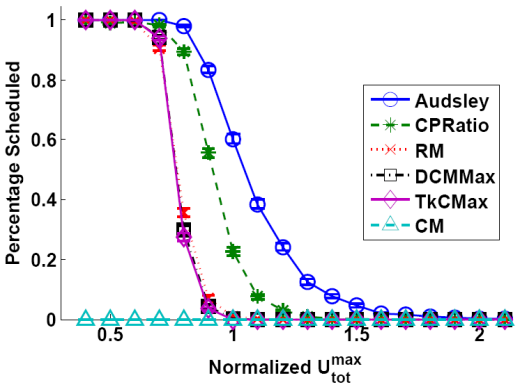
We developed a simulator in the Java language that generates synthetic, mixed-criticality task sets with several parameters varied over a wide range in an effort to represent a variety of hard real-time applications. Our simulation implements the *UUnifast-Discard* algorithm described by Davis and Burns (2009) for the generation of task utilisations. The value generated by UUnifast-Discard for a task τ_i is assigned as the value for $u_i(k)$ – the utilisation of τ_i at the highest criticality level. For each task, a utilisation function is generated for the criticality levels of the task set with values uniformly distributed in $[0.4u_i(k), u_i(k)]$. Periods are generated randomly between a minimum period of 10 and a maximum period of 1,000 milliseconds. The criticality levels of the tasks in a set are uniformly distributed in $[1, k]$.

Task sets were generated with U_{max}^{tot} values varied between 0.8 m and 3.0 m, where m is the number of processors. For each of a selected set of values in this range, 1,000 task sets were generated with varying values for the number of tasks in a set and the number of criticality levels. Task sets were generated with 2, 4, and 8 criticality levels and with 40 and 60 tasks. We considered 2-processor and 4-processor systems.

We implemented the schedulability test proposed by Bertogna et al. (2009), which we refer to as B2009, and the test described by Bertogna and Cirinei (2007), which we refer to as BC2007. B2009 and BC2007 were implemented in accordance with equations (1) through (3), applied in the context of the mixed-criticality model proposed by Vestal (2007). In particular, when evaluating the schedulability of a task τ_k, the WCET used for each higher-priority task τ_i is the WCET of τ_i corresponding to the criticality level of τ_k. We implemented the priority-assignment algorithms RM, CM, TkC, DCM, and Audsley's algorithm, along with our proposal (*CPRatio*), all of which were described in Section 5. With TkC and DCM, the priority assigned to a task is dependent in part on its WCET. For both, we have chosen to use the WCET associated with the highest criticality level for all tasks. We refer to these algorithms, when used with this WCET selection, as TkCMax and DCMMax, respectively. With two schedulability tests (B2009 and BC2007) and these six priority-assignment strategies, we have a total of 12 combinations of schedulability test and priority-assignment algorithm. We assessed the schedulability of every generated task set under these 12 combinations.

Results are shown in Figure 4 for sets of 40 tasks with 4 criticality levels, scheduled on 4 CPUs using B2009. The fraction of task sets deemed schedulable by B2009 is plotted as a function of normalised U_{max}^{tot}. With these parameters, Audsley's algorithm offers significantly better performance relative to the other priority assignment schemes. With *CPRatio*, fewer task sets are deemed schedulable relative to the results from Audsley's algorithm. However, *CPRatio* provides a significant increase in performance over the other priority-assignment approaches, and this increase is achieved without the significant complexity of Audsley's algorithm. As shown in Figure 4, RM, DCMMax, and TkCMax provide the same level of performance in this setting. In addition, very few task sets are schedulable with CM priority assignment.

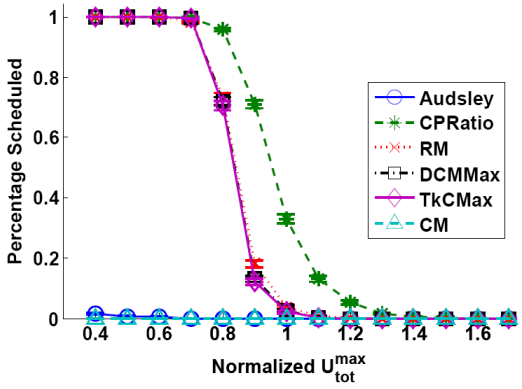
Figure 4 4 Criticality levels, 40 tasks, 4 CPUs, B2009 feasibility test (see online version for colours)



In Figure 5, results are shown for the same parameters used to generate Figure 4, except that schedulability is determined using BC2007. A striking observation in this case is that almost no task sets are schedulable when priorities are assigned by Audsley's algorithm. This is due to the fact, as shown by Davis and Burns (2011), that Audsley's algorithm is not compatible with BC2007. In Figure 5, *CPRatio* provides significantly improved performance relative to the other priority-assignment schemes. While the relative performance of the algorithms

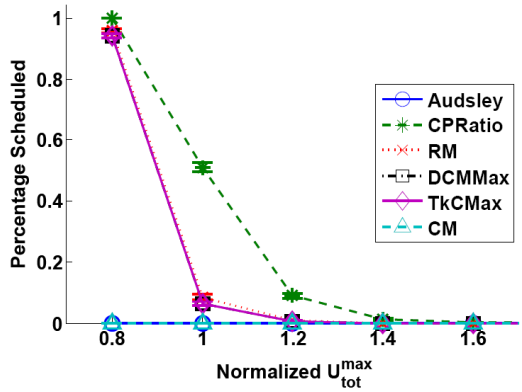
as depicted in Figure 5 (except for Audsley's algorithm) is the same as that shown in Figure 4, the absolute performance of each algorithm is better in Figure 5. This is expected given the tighter upper bound on workload used by BC2007. We note that, while the performance of *CPRatio* is improved with BC2007 relative to B2009, it offers lower performance than the combination of Audsley's algorithm with B2009.

Figure 5 4 Criticality levels, 40 tasks, 4 CPUs, BC2007 feasibility test (see online version for colours)



Results are shown in Figure 6 for the same parameters used to generate Figure 5, except that tasks are scheduled on two CPUs rather than 4. The main trends remain the same with this reduction in the number of CPUs. However, there is a small increase in the absolute performance of each algorithm. Since the results are plotted for normalised U_{max}^{tot} values, the tasks represented in Figure 6 for a given value of U_{max}^{tot} tend to have smaller utilisation values relative to those of Figure 5. With smaller utilisation values, the negative impact of *criticality inversions* (de Niz et al., 2009), where a high-criticality task is given lower priority relative to a low-criticality task, is typically reduced, resulting in an increase in schedulability.

Figure 6 4 Criticality levels, 40 tasks, 2 CPUs, BC2007 feasibility test (see online version for colours)



In Figures 7 through 9, results are shown for the same parameters used to generate Figures 4 through 6, respectively, except that task sets are comprised of 60 tasks rather than 40. As is evident in the plots, the main trends remain the same.

Figure 7 4 Criticality levels, 60 tasks, 4 CPUs, B2009 feasibility test (see online version for colours)

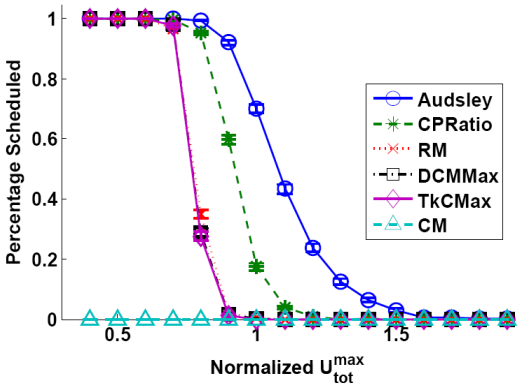


Figure 8 4 Criticality levels, 60 tasks, 4 CPUs, BC2007 feasibility test (see online version for colours)

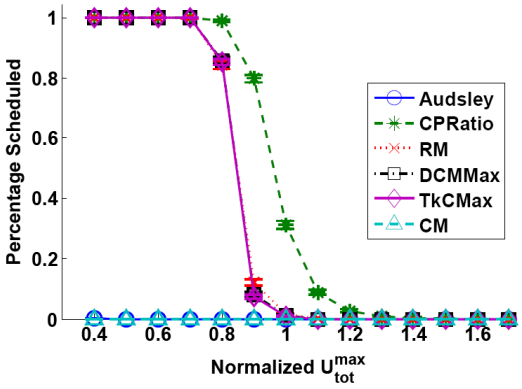
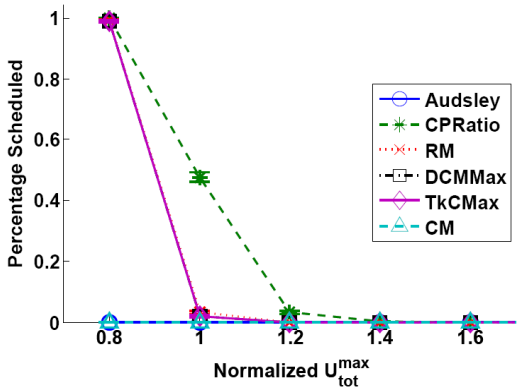


Figure 9 4 Criticality levels, 60 tasks, 2 CPUs, BC2007 feasibility test (see online version for colours)



Results are presented in Figures 10 through 12 as a function of the number of criticality levels. Figure 10 depicts results for 40 tasks scheduled on 4 CPUs with schedulability evaluated by B2009. Results are plotted for a U_{max}^{tot} value of 1.0. The performance of both Audsley's algorithm and *CPRatio* increases with an increase in the number of criticality levels. With more criticality levels, WCETs are assigned with a finer degree of granularity. As a result, in cases in which a high-criticality task is assigned a lower priority relative to a low-criticality task (i.e., a criticality inversion) the increase in the WCET attributed to the lower-criticality task relative to its nominal

WCET (i.e., that corresponding to its own criticality) is typically smaller, resulting in a smaller negative impact to the schedulability of the task set. With these parameters, the number of task sets schedulable with priorities assigned by Audsley's algorithm is significantly higher than that for the other priority-assignment strategies. *CPRatio* offers performance that is significantly lower than that of Audsley's algorithm, but significantly higher than the other algorithms. In fact, very few task sets are schedulable by RM, DCMax, TkCMax, or CM.

Figure 10 40 Tasks, 4 CPUs, B2009 feasibility test (see online version for colours)

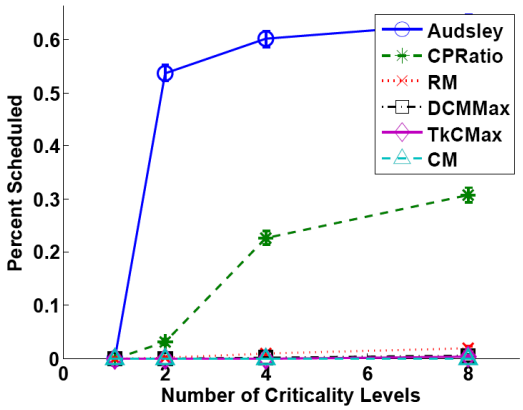


Figure 11 40 Tasks, 4 CPUs, BC2007 feasibility test (see online version for colours)

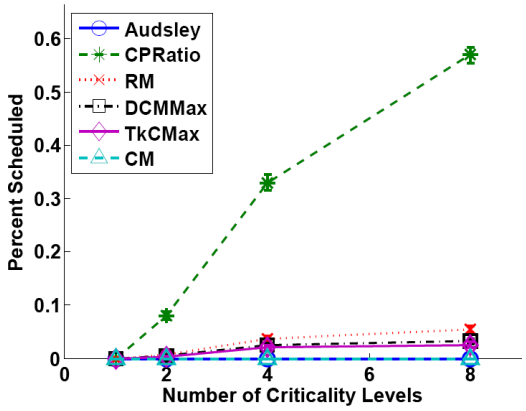
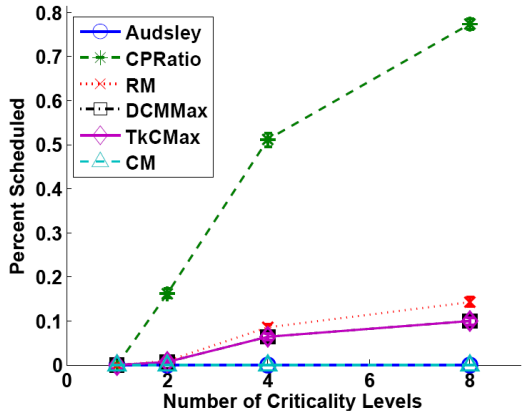


Figure 12 40 Tasks, 2 CPUs, BC2007 feasibility test (see online version for colours)



In Figure 11, results are shown for the same parameters used to generate the results shown in Figure 10, except that schedulability is determined by BC2007. As before, the performance of Audsley's algorithm is very poor with BC2007. The relative ordering of the other schemes remains the same. Absolute results improve slightly with the change from B2009 to BC2007. This is due to the tighter workload bound provided by BC2007. Also, as was the case in Figure 10, performance increases with an increase in the number of criticality levels.

In Figure 12, results are shown for the same parameters used to generate the results in Figure 11, except that tasks are scheduled on 2 CPUs rather than 4. The relative performance of the algorithms remains the same with this change. However, a small increase in the absolute performance of the algorithms is noticeable with this decrease in the number of CPUs. This is consistent with the improvement shown with a reduction in the number of CPUs in Figure 6 relative to Figure 5 and may be explained by a reduction in the impact of criticality inversions associated with lower-utilisation tasks.

7 Conclusions

In this paper, we have investigated the use of global FP algorithms for the scheduling of mixed-criticality task sets on multiprocessors. We have explored the two key dimensions of this problem – priority assignment and schedulability testing. We have examined state-of-the art algorithms developed in each of these dimensions for traditional task sets and have performed an experimental evaluation of their performance in the mixed-criticality context using the model proposed by Vestal (2007). We have also proposed and evaluated a new priority-assignment algorithm called *CPRatio* that takes both period and criticality into account.

Our experimental results show that Audsley's priority assignment algorithm when combined with the schedulability test proposed by Bertogna et al. (2009) (referred to as B2009) outperforms other approaches. We have also shown that CPRatio performs significantly better than all other priority-assignment schemes (i.e., except for Audsley's algorithm). This is an important observation for cases in which the simplicity of CPRatio is preferred over the increased complexity of Audsley's algorithm. We have also shown that CPRatio, when combined with BC2007, offers significantly better performance relative to other priority assignment schemes, including Audsley's algorithm. This suggests that our algorithm, CPRatio, is the only algorithm that offers robust performance under both global schedulability tests.

Acknowledgements

This work was, in part, supported by US National Science Foundation award CNS-1016855.

References

Andersson, B. and Jonsson, J. (2000) 'Some insights on fixed-priority preemptive non-partitioned multiprocessor scheduling', in *Proceedings of the 21st IEEE Real-Time Systems Symposium, Work-in-Progress Session*.

Andersson, B. and Jonsson, J. (2003) 'The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%', in *Proceedings of the 15th Euromicro Conference on Real-Time Systems*.

Audsley, N.C. (1991) *Optimal Priority Assignment and Feasibility of Static Priority Tasks with Arbitrary Start Times*, Technical report, University of York, England, November.

Baker, T.P. (2003) 'Multiprocessor EDF and deadline monotonic schedulability analysis', n *Proceedings of the 24th IEEE Real-Time Systems Symposium*.

Baruah, S. and Vestal, S. (2008) 'Schedulability analysis of sporadic tasks with multiple criticality specifications', in *Proceedings of the 20th Euromicro Conference on Real-Time Systems*.

Baruah, S., Bonifaci, V., D'Angelo, G., Li, H., Marchetti-Spaccamela, A., Megow, N. and Stougie, L. (2012) 'Scheduling real-time mixed-criticality jobs', in *IEEE Transactions on Computers*, August, Vol. 61, No. 8, pp.1140–1152.

Baruah, S., Li, H. and Stougie, L. (2010) 'Towards the design of certifiable mixed-criticality systems', in *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*.

Bertogna, M. and Cirinei, M. (2007) 'Response-time analysis for globally scheduled symmetric multiprocessor platforms', in *Proceedings of the 28th IEEE Real-Time Systems Symposium*.

Bertogna, M., Cirinei, M. and Lipari, G. (2009) 'Schedulability analysis of global scheduling algorithms on multiprocessor platforms', *IEEE Transactions on Parallel and Distributed Systems*, April, Vol. 20, No. 4, pp.553–566.

Davis, R.I. and Burns, A. (2009) 'Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems', in *Proceedings of the 30th IEEE Real-Time Systems Symposium*.

Davis, R.I. and Burns, A. (2011) 'Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems', *Real-Time Systems*, January, Vol. 47, No. 1, pp.1–40.

de Niz, D., Lakshmanan, K. and Rajkumar, R. (2009) 'On the scheduling of mixed-criticality real-time task sets', in *Proceedings of the 30th IEEE Real-Time Systems Symposium*.

Dhall, S.K. and Liu, C.L. (1978) 'On a real-time scheduling problem', *Operations Research*, February, Vol. 26, No. 1, pp.127–140.

Guan, N., Stigge, M., Yi, W. and Yu, G. (2009) 'New response time bounds for fixed priority multiprocessor scheduling', in *Proceedings of the 30th IEEE Real-Time Systems Symposium*.

Jersak, M., Richter, K. and Ernst, R. (2005) 'Performance analysis for complex embedded applications', *International Journal of Embedded Systems*, Vol. 1, No. 1, pp.33–49.

Joseph, M. and Pandya, P. (1986) 'Finding response times in a real-time system', *The Computer Journal*, October, Vol. 29, No. 5, pp.390–395.

Kelly, O.R., Aydin, H. and Zhao, B. (2011) 'On partitioned scheduling of fixed-priority mixed-criticality task sets', in *Proceedings of the IEEE International Conference on Embedded Software and Systems*.

Lakshmanan, K., de Niz, D. and Rajkumar, R. (2011) 'Mixed-criticality task synchronization in zero-slack scheduling', in *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium*.

Lakshmanan, K., de Niz, D., Rajkumar, R. and Moreno, G. (2010) 'Resource allocation in distributed mixed-criticality cyber-physical systems', in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems*.

Lauzac, S., Melhem, R. and Mosse, D. (1998) 'Comparison of global and partitioned schemes for scheduling rate monotonic tasks on a multiprocessor', in *Proceedings of the Euromicro Workshop on Real-Time Systems*.

Leung, J.Y-T. and Whitehead, J. (1982) 'On the complexity of fixed-priority scheduling of periodic, real-time tasks', *Performance Evaluation*, December, Vol. 2, No. 4, pp.237–250.

Li, H. and Baruah, S. (2010) 'Load-based schedulability analysis of certifiable mixed-criticality systems', in *Proceedings of the 10th ACM International Conference on Embedded Software.*

Liu, C.L. and Layland, J.W. (1973) 'Scheduling algorithms for multiprogramming in a hard-real-time environment', *Journal of the ACM*, January, Vol. 20, No. 1, pp.46–61.

Liu, J.W.S. (2000) *Real-Time Systems*, Prentice Hall, Upper Saddle River, NJ.

Lopez, J.M., Diaz, J.L. and Garcia, D.F. (2004) 'Utilization bounds for EDF scheduling on real-time multiprocessor systems', *Real-Time Systems*, October, Vol. 28, No. 1, pp.39–68.

Lopez, J.M., Garcia, M., Diaz, J.L. and Garcia, D.F. (2003) 'Utilization bounds for multiprocessor rate-monotonic systems', *Real-Time Systems*, January, Vol. 24, No. 1, pp.5–28.

Mollison, M.S., Erickson, J.P., Anderson, J.H., Baruah, S.K. and Scoredos, J.A. (2010) 'Mixed-criticality real-time scheduling for multicore systems', in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology*.

Pathan, R.M. (2012) 'Schedulability analysis of mixed-criticality systems on multiprocessors', in *Proceedings of the 24th Euromicro Conference on Real-Time Systems*.

Vestal, S. (2007) 'Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance', in *Proceedings of the 28th IEEE International Real-Time Systems Symposium*.