FPZL Schedulability Analysis

Robert I. Davis and Alan Burns
Real-Time Systems Research Group, Department of Computer Science,
University of York, YO10 5DD, York (UK)
rob.davis@cs.york.ac.uk, alan.burns@cs.york.ac.uk

Abstract

This paper presents the FPZL scheduling algorithm for multiprocessor real-time systems. FPZL is similar to global fixed priority pre-emptive scheduling; however, whenever a task reaches a state of zero laxity it is given the highest priority. FPZL is a minimally dynamic algorithm, in that the priority of a job can change at most once during its execution.

Polynomial time and pseudo-polynomial time sufficient schedulability tests are derived for FPZL. These tests are then improved by computing upper bounds on the amount of execution that each task can perform in the zero laxity state. An empirical evaluation shows that FPZL is highly effective, with a significantly larger number of tasksets deemed schedulable by the tests derived in this paper, than by state-of-the-art schedulability tests for EDZL scheduling.

1. Introduction

In this paper, we introduce a minimally dynamic global scheduling algorithm for real-time multiprocessor systems called FPZL (Fixed Priority until Zero Laxity). FPZL is based on global fixed priority pre-emptive scheduling, which for brevity we refer to as global FP scheduling. Under FPZL, jobs are scheduled according to the fixed priority of their associated task, until a situation is reached where the remaining execution time of a job is equal to the time to its deadline. Such a job has *zero laxity* and will miss its deadline unless it executes continually until completion. FPZL gives such zero laxity jobs the highest priority. The schedules produced by FPZL and global FP scheduling are identical until the latter fails to execute a task with zero laxity. Such a task will subsequently miss its deadline. Hence FPZL dominates global FP scheduling, in the sense that all tasksets / priority ordering combinations that are schedulable according to global FP scheduling are also schedulable according to FPZL. FPZL is closely related to EDZL [30], [21], [7], [20], [32], [33], [23] which applies the same zero laxity rule to global EDF scheduling.

1.1. Related work

For a comprehensive survey of multiprocessor real-time scheduling, including early work on utilisation-based schedulability tests for global FP, and global EDF scheduling of periodic tasksets with implicit deadlines, the interested reader is referred to [25].

During the last ten years, sophisticated schedulability tests have been developed for global FP, and global EDF scheduling of sporadic tasksets with constrained and arbitrary deadlines. These tests rely on the analysis of response times and processor load rather than utilisation.

In 2000, Andersson and Jonsson [1] provided a simple response time test applicable to tasksets with constrained-deadlines scheduled using global FP scheduling.

In 2003, Baker [4] developed a fundamental schedulability test strategy, based on considering the minimum amount of interference in a given interval that is necessary to cause a deadline to be missed, and then taking the contra-positive of this to form a sufficient schedulability test. This basic strategy underpins an extensive thread of subsequent research into schedulability tests for global EDF [8], [16], [13], [12], global FP [11], [17], [5], [26], and EDZL scheduling [21].

Baker's work was subsequently built upon by Bertogna et al. [14] in 2005, (see also Bertogna and Cirinei [17]). They developed sufficient schedulability tests for: (i) any work conserving algorithm, (ii) global EDF, and (iii) global FP scheduling based on bounding the maximum workload in a given interval. In 2007, Bertogna and Cirinei [15] adapted this approach to iteratively compute an upper bound on the response time of each task, using the upper bound response times of other tasks to limit the amount of interference considered. In 2009, Guan et al. [28] extended the response time analysis of Bertogna and Cirinei [15] for global FP scheduling, using ideas from [10].

In 2009, Davis and Burns [24] showed that priority assignment is fundamental to the effectiveness of global FP scheduling. They proved that Audsley's optimal priority assignment algorithm [2], [3] is applicable to some of the sufficient tests developed for global FP scheduling, including the simple response time test of Andersson and Jonsson [1] and the deadline-based test of Bertogna et al. [17], but not to others such as the response time tests of Bertogna and Cirinei [15], and Guan et al. [28].

The Earliest Deadline first until Zero Laxity (EDZL) algorithm was introduced in 1994 by Lee [30], who showed that EDZL dominates global EDF scheduling, and is sub-optimal for two processors (see also Cho et al. [23], Park et al. [33]). Here, sub-optimal, is used to mean that EDZL can "schedule any feasible set of ready tasks". This weak form of optimality is appropriate for online scheduling algorithms, which cannot take account of future arrival times. In 2006, Piao et al. [32] showed that EDZL is completion time predictable in the sense defined by Ha and Liu [29], (see Section 2.1). A simpler proof of predictability was given by Cirinei and Baker [21] in 2007, who also developed a sufficient schedulability test for EDZL based on the fundamental strategy of Baker [4].

In 2008, Baker et al [7] gave an iterative sufficient test for EDZL based on the approach taken by Bertogna et al. [17] for work conserving algorithms and global EDF. This test reduces the over-estimation of carry-in interference, a feature of the previous tests, by iteratively calculating a lower bound on the slack for each task. The empirical evaluation in [7] shows that this iterative test for EDZL outperforms other tests for EDZL given in [21] and, as expected, similar tests for global EDF.

Also in 2008, Kato and Yamasaki [31], introduced EDCL, effectively a variant of EDZL, which increases job priority on the basis of critical-laxity at the release or completion time of a job. This has the effect of reducing the maximum number of context switches to two per job, the same as EDF, at the expense of slightly inferior schedulability, when compared to EDZL. Kato and Yamasaki [31] also corrected a minor flaw in the polynomial time schedulability test for EDZL given in [21].

1.2. Intuition and motivation

The research described in this paper is motivated by the need to close the large gap that currently exists between the best known approaches to multiprocessor real-time scheduling for sporadic tasksets with constrained deadlines and what may be possible as indicated by feasibility / infeasibility tests.

Dynamic priority scheduling has the potential to schedule many more tasksets than fixed task or fixed job priority algorithms. However, this theoretical advantage must be balanced against the increased overheads inherent in dynamic changes in priority. For example, the LLREF scheduling algorithm [22], which is optimal for periodic tasksets with implicit deadlines, and the LRE-TL scheduling algorithm [27] which is optimal for sporadic tasksets with implicit deadlines, divide the timeline into intervals that start and end at task releases/deadlines (referred to as TL-planes in [22]). In each interval, LLREF and LRE-TL ensure that each active task τ_i executes for at least $U_i t$, where U_i is the task's utilisation, and *t* is the length of the time interval. Hence every task can in the worst-case execute in every interval between task deadlines, resulting in *n*-1 pre-emptions per job release, where *n* is the number of tasks. In systems with a large number of tasks, this level of pre-emptions leads to prohibitively high overheads.

Minimally dynamic scheduling algorithms, such as FPZL (and EDZL) offer a potential solution to this problem. Note, by *minimally dynamic*, we mean that the priority of a job changes at most once during its execution, hence bounding the number of pre-emptions to at most two per job release.

1.3. Organisation

The remainder of the paper is organised as follows: Section 2 describes the terminology, notation and system model used. Section 3 describes sufficient tests for global FP scheduling. These tests are used in Section 4 to derive polynomial time and pseudo-polynomial time sufficient schedulability tests for FPZL. Section 5 shows how the schedulability tests for FPZL can be improved by bounding the amount of execution that each task can perform in the zero laxity state. Section 6 provides a brief discussion of priority assignment for FPZL. Section 7 presents an empirical investigation into the effectiveness of FPZL and its schedulability tests. Finally, Section 8 concludes with a summary and suggestions for future research.

2. System model, terminology and notation

In this paper, we are interested in FPZL scheduling of an application on a homogeneous multiprocessor system comprising *m* identical processors. The application or taskset is assumed to comprise a static set of *n* tasks ($\tau_1 ... \tau_n$), where each task τ_i is assigned a unique priority *i*, from 1 to *n* (where *n* is the lowest priority).

Tasks are assumed to comply with the *sporadic* task model. In this model, tasks give rise to a potentially infinite sequence of jobs. Each job of a task may arrive at any time once a minimum inter-arrival time has elapsed since the arrival of the previous job of the same task.

Each task τ_i is characterised by its relative *deadline* D_i, *worst-case execution time* C_i, and minimum inter-arrival time or *period* T_i. The *utilisation* U_i of each task is given by C_i / T_i. A task's *worst-case response time* R_i is defined as the longest time from a job of the task arriving to it completing execution.

It is assumed unless otherwise stated that all tasks have constrained deadlines ($D_i \le T_i$). The tasks are assumed to be independent and so cannot be blocked from executing by another task other than due to contention for the processors. Further, it is assumed that once a task starts to execute it will not voluntarily suspend itself.

Intra-task parallelism is not permitted; hence, at any given time, each job may execute on at most one processor. As a result of pre-emption and subsequent resumption, a job may migrate from one processor to another. The cost of pre-emption, migration, and the run-time operation of the scheduler is assumed to be either negligible, or subsumed into the worst-case execution time of each task.

Under global FP scheduling, at any given time, the *m* highest priority ready jobs are executed. Under FPZL scheduling, if a job reaches a state of zero laxity, then it is given the highest priority and will execute until completion. The laxity of a job is given by the elapsed time to its deadline less its remaining execution time. Under FPZL, at any given time, at most *m* tasks may be in the zero laxity state without a deadline being missed.

Schedulability analysis may identify certain tasks as being able to enter the zero laxity state. We refer to these tasks as *zero laxity tasks*. An upper bound on the maximum amount of execution that a job of task τ_i can perform in the zero laxity state is denoted by Z_i^{UB}.

The following notation is used to refer to subsets of tasks: *hp*(*i*) is the set of tasks with priorities higher than *i*, and *lpzl*(*i*) is the set of zero laxity tasks with initial priorities lower than *i*.

2.1. Predictability and sustainability

In 1994, Ha and Liu [29] defined the concept of scheduling algorithm *predictability*. A scheduling algorithm is referred to as *predictable* if the response times of jobs cannot be increased by decreases in their execution times, with all other parameters remaining constant. Predictability is a fundamental requirement for scheduling algorithms as in real systems task execution times are almost always variable up to some worst-case bound.

Ha and Liu [29] proved that all priority driven (i.e. fixed job priority and fixed task priority) pre-emptive scheduling algorithms for multiprocessor systems are predictable. However, FPZL is a dynamic priority algorithm, and as such it is necessary to prove the predictability of FPZL before it can be considered useful.

Theorem 1: (Predictability of FPZL) The FPZL scheduling algorithm is predictable with respect to decreases in task execution times.

Proof sketch: Proof follows the logic used in Theorem 1 of [7] to prove that EDZL is predictable. Noting that under FPZL, the choice to schedule different jobs from the two sets of jobs considered in Theorem 1 of [7] can only be due to the zero laxity rule. This is because the jobs belong to the same task and therefore have the same priority. They differ only in their execution times.

For completeness, the definition of predictability and the proof given in [7] are reproduced below, the latter modified so that it applies to FPZL rather than EDZL.

Definition: Completion-time *predictability* (reproduced from [7]): *A scheduling algorithm is defined to be completion-time predictable if, for every pair of sets J and J' of jobs that differ only in the execution times of the jobs, and such that the execution times of jobs in J' are less than or equal to the execution times of the corresponding jobs in J, then the completion time of each job in J' is no later than the completion time of the corresponding job in J.*

Proof: (Reproduced from [7], and modified (underlined text) to apply to FPZL).

We actually prove a stronger hypothesis; that is, if the only difference between J and J' is that some of the actual job execution times are shorter in J' than in J, then the accumulated execution time of every uncompleted job in the <u>FPZL</u> schedule for J' is greater than or equal to the accumulated execution time of the same job in the <u>FPZL</u> schedule for J at every instant in time. It will follow that no job can have an earlier completion time in J than in J', since the actual execution times in J are at least as long as in J'. Suppose the above hypothesis is false. That is, there exist job sets J and J' whose only difference is that some of the actual job execution times are shorter in J' than in J, and such that at some time t the accumulated execution time of some uncompleted job is less with J' than with J. We will show that this leads to a contradiction, and the theorem will follow.

Without loss of generality, we can restrict attention to the case where J and J' differ only in the actual execution time of one job. To see this, observe that between J and J' there is a finite sequence of sets of jobs such that the only difference between one set and the next is that the actual execution time of one job is decreased. Let J and J' be the first pair [of sets] in such a sequence such that at some time t the accumulated execution time of some uncompleted job j is less with J' than with J.

Let t be the earliest instant in time after which the accumulated execution time of some uncompleted job is less with J' than with J, and let j be such a job. That is, up through t the accumulated execution time of each uncompleted job in the schedule for J is less than or equal to the accumulated execution time of the same job in the schedule for J', and after time t the accumulated execution time of job j is greater with J than with J'.

Job j must be scheduled to execute starting at time t with J and not with J'. This means some other job j' is scheduled to execute in place of j with J'. That choice cannot be based on <u>priority</u>, since the <u>priorities</u> of corresponding jobs are the same with J and J', so it must be based on the zero-laxity rule. That is, j' has zero laxity at time t with J' but not with J. However, that would require that j' has greater accumulated execution time at time t with J than it does with J'. This is a contradiction of the choice of t □

3. Schedulability tests for global FP

In this section, we outline two sufficient schedulability tests for global FP scheduling of sporadic tasksets. The first was originally developed by Bertogna et al [17] and uses deadline analysis. We refer to this test as the "DA test". The second was developed by Bertogna and Cirinei [15] and uses response time analysis. We refer to this test as the "RTA test". Subsequently, the RTA test was improved by Guan et al. [28], using ideas from [10] which limit the amount of carry-in interference. We refer to this improved test as the "RTA-LC test" (Response Time Analysis with Limited Carry-in). The approach of Guan et al. is also applicable to the deadline analysis used in [17], and so we also describe an improved version of the DA test that also limits carry-in. We refer to that improved test as the "DA-LC test" (Deadline Analysis with Limited Carry-in). Each of the tests described is based on the fundamental strategy derived by Baker [4].

3.1. Deadline Analysis for global FP

In [17], Bertogna et al. developed a polynomial time sufficient schedulability test for global FP scheduling based on deadline analysis. They showed that if task τ_k is schedulable in an interval of length L, then an upper bound on the interference over the interval due to a higher priority task τ_i with a carry-in job (released prior to the start of the interval) is given by[1]:

$$I_i^D(L) = \min(W_i^D(L), L - C_k + 1) \quad (1)$$

where $W_i^D(L)$ is an upper bound on the workload of task τ_i in an interval of length L, given by:

$$W_i^D(L) = N_i^D(L)C_i + \min(C_i, L + D_i - C_i - N_i^D(L)T_i) \quad (2)$$

and $N_i^D(L)$ is the maximum number of jobs of task τ_i that contribute all of their execution time in the interval:

[1] Note we adopt the approach to time representation used in [17]. Time is represented by non-negative integer values, with each time value *t* viewed as representing the whole of the interval [*t*, *t*+1). This enables mathematical induction on clock ticks and avoids confusion with respect to end points of execution.

$$N_i^D(L) = \left\lfloor \frac{L + D_i - C_i}{T_i} \right\rfloor \tag{3}$$

Bertogna et al. [17] used Equation (1), with D_k as the length of the interval, and strategy of Baker [4] to form a schedulability test for each task τ_k:

DA test for global FP scheduling: A sporadic taskset is schedulable, if for every task τ_k in the taskset, the inequality given by Equation (4) holds:

$$D_k \geq C_k + \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_i^D(D_k) \right\rfloor \tag{4}$$

where *hp*(*k*) is the set of tasks with priorities higher than *k*. Note we have re-written Equation (4) in a different form from that presented in [17] for ease of comparison with the response time schedulability test given in [15].

Guan et al. [28] showed that if task τ_k is schedulable in an interval of length *L*, then an upper bound on the interference over the interval due to a higher priority task τ_i *without* a carry in job is given by:

$$I_i^{NC}(L) = \min(W_i^{NC}(L), L - C_k + 1) \tag{5}$$

where:

$$W_i^{NC}(L) = N_i^{NC}(L)C_i + \min(C_i, L - N_i^{NC}(L)T_i) \tag{6}$$

and

$$N_i^{NC}(L) = \lfloor L / T_i \rfloor \tag{7}$$

The difference between the two interference terms given by Equations (1) and (5) is:

$$I_i^{DIFF-D}(L) = I_i^D(L) - I_i^{NC}(L) \tag{8}$$

Applying the approach of Guan et al. [28], an improved version of the DA test can be stated as follows:

DA-LC test for global FP scheduling: A sporadic taskset is schedulable, if for every task τ_k in the taskset, the inequality given by Equation (9) holds:

$$D_k \geq C_k + \left\lfloor \frac{1}{m} \left(\sum_{\forall i \in hp(k)} I_i^{NC}(D_k) + \sum_{i \in MD(k,m-1)} I_i^{DIFF-D}(D_k) \right) \right\rfloor \tag{9}$$

where *MD*(*k*, *m*-1) is the subset of the *min*(*k*, *m*-1) tasks with the largest values of $I_i^{DIFF-D}(D_k)$ from the set of tasks *hp*(*k*).

We note that the DA-LC test reduces to the DA test if the $I_i^{DIFF-D}(D_k)$ term is included for all of the higher priority tasks, rather than just those with the *m*-1 largest values, hence the DA-LC test dominates the DA test.

3.2. Response Time Analysis for global FP

Bertogna and Cirinei [15] extended the basic approach used in the DA test to iteratively compute an upper bound response time R_k^{UB} for each task, using the upper bound response times of higher priority tasks to limit the amount of interference considered. This approach applies the same logic as [17], while recognising that the latest time that a task can execute is when it completes with its worst-case response time rather than at its deadline.

In [15], Bertogna and Cirinei showed that if task τ_k is schedulable in an interval of length *L*, then an upper bound on the interference in that interval due to a higher priority task τ_i with a carry-in job is given by:

$$I_i^R(L) = \min(W_i^R(L), L - C_k + 1) \tag{10}$$

where, $W_i^R(L)$ is an upper bound on the workload of task τ_i in an interval of length *L*, taking into account the upper bound response time of task τ_i:

$$W_i^R(L) = N_i^R(L)C_i + \min(C_i, L + R_i^{UB} - C_i - N_i^R(L)T_i) \tag{11}$$

and $N_i^R(L)$ is given by:

$$N_i^R(L) = \left\lfloor \frac{L + R_i^{UB} - C_i}{T_i} \right\rfloor \tag{12}$$

The response time test of Bertogna and Cirinei [15] may be expressed as follows:

RTA test for global FP scheduling (Theorem 7 in [15]): A sporadic taskset is schedulable, if for every task τ_k in the taskset, the upper bound response time R_k^{UB} computed via the fixed point iteration given in Equation (13) is less than or equal to the task's deadline:

$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_i^R(R_k^{UB}) \right\rfloor \tag{13}$$

Iteration starts with $R_k^{UB} = C_k$, and continues until the value of R_k^{UB} converges or until $R_k^{UB} > D_k$, in which case task τ_k is unschedulable.

We note that using the RTA test, task schedulability needs to be determined in priority order, highest priority first, as upper bounds on the response times of higher priority tasks are required for computation of the interference term $I_i^R(R_k^{UB})$.

In [28], Guan et al. showed that at most *m*-1 higher priority tasks with carry-in jobs may contribute interference in the worst-case, and used this result to improve the RTA test as follows:

Guan et al. [28] showed that if task τ_i does not have a carry-in job, then the interference term is given by Equation (5). The difference between the two interference terms (Equation (10) and Equation (5)) is then given by:

$$I_i^{DIFF-R}(L) = I_i^R(L) - I_i^{NC}(L) \tag{14}$$

Using this result, Guan et al. [28] improved upon the response time test of Bertogna and Cirinei [15].

RTA-LC test for global FP scheduling: A sporadic taskset is schedulable, if for every task τ_k in the taskset, the upper bound response time R_k^{UB} computed via the fixed point iteration given in Equation (15) is less than or equal to the task's deadline:

$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m} \left(\sum_{\forall i \in hp(k)} I_i^{NC}(R_k^{UB}) + \sum_{i \in MR(k,m-1)} I_i^{DIFF-R}(R_k^{UB}) \right) \right\rfloor \tag{15}$$

where *MR*(*k*, *m*-1) is the subset of the *min*(*k*, *m*-1) tasks with the largest values of $I_i^{DIFF-R}(R_k^{UB})$, given by Equation (14), from the set of tasks *hp*(*k*). Iteration starts with $R_k^{UB} = C_k$, and continues until the value of R_k^{UB} converges or until $R_k^{UB} > D_k$, in which case task τ_k is unschedulable.

We note that the RTA-LC test reduces to the RTA test if the $I_i^{DIFF-R}(R_k^{UB})$ term is included for all of the higher priority tasks, rather than just those with the m-1 largest values, hence the RTA-LC test dominates the RTA test. Both the RTA and RTA-LC tests for global FP scheduling are pseudo-polynomial in complexity.

4. Schedulability tests for FPZL

In this section, we derive polynomial time and pseudo-polynomial time sufficient schedulability tests for FPZL. These tests are applicable to sporadic tasksets with constrained deadlines, and are independent of the priority assignment policy used. They are based on the tests described in the previous section for global FP scheduling.

4.1. Deadline Analysis for FPZL

Schedulability under FPZL differs from that under global FP scheduling in two important aspects:

1. Under FPZL, up to m tasks may be deemed unschedulable according to analysis of their response times; and yet, due to the zero laxity rule, the tasks will not miss their deadlines.
2. Tasks that reach the zero laxity state have an additional impact on the schedulability of other tasks.

We now derive the maximum interference on a higher priority task τ_k, in a problem window of length L, that could possibly be caused by a lower priority task τ_j executing in the zero laxity state.

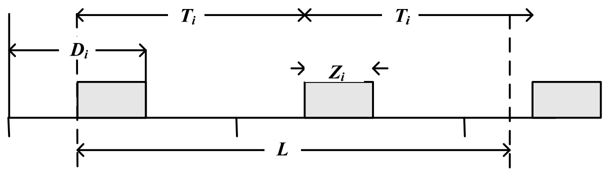

Figure 1: Interference in an interval

Figure 1 illustrates the worst-case scenario. This occurs when the first job of τ_j in the problem window starts executing in the zero laxity state, i.e. at the highest priority, at the start of the problem window, and completes at its deadline. As zero laxity execution can only occur immediately prior to a task's deadline, subsequent zero laxity execution of the jobs of τ_j occurs at minimum intervals of T_i. Thus an upper bound on the amount of zero laxity workload due to task τ_j in an interval of length L is given by:

$$W_j^Z(L) = N_j^Z(L) Z_j^{UB} + \min(Z_j^{UB}, L - N_j^Z(L) T_j) \qquad (16)$$

where $N_j^Z(L)$ is the maximum number of jobs of task τ_j that contribute all of their zero laxity execution in the interval

$$N_j^Z(L) = \lfloor L / T_j \rfloor \qquad (17)$$

Note, Z_j^{UB} $(\le C_j)$ is an upper bound on the amount of execution that any job of task τ_j can perform in the zero laxity state.

If task τ_k is schedulable in an interval of length L, then an upper bound on the interference in that interval due to a lower priority task τ_j executing in the zero laxity state is given by:

$$I_j^Z(L) = \min(W_j^Z(L), L - C_k + 1) \qquad (18)$$

We observe that interference on task τ_k due to a lower priority zero laxity task τ_j, in an interval of length L, is the same irrespective of whether τ_j is considered as having a carry-in job or not. Hence task τ_j contributes zero additional carry-in interference and so does not need to be included when determining the m-1 tasks that contribute the largest amounts of additional carry-in interference (i.e. the I_i^{DIFF-D} terms – see Equation (8)).

We now consider the interference from a higher priority task τ_i capable of entering the zero laxity state. In this case, the maximum interference with a carry-in job occurs when the first job of τ_i in the problem window starts executing at the start of the problem window, and completes at its deadline, with all subsequent jobs executing as early as possible, see Figure 2 below. We observe that this is effectively the same scenario that leads to the worst-case interference from a higher priority task which does not enter the zero laxity state but completes at its deadline, and is given by Equation (1). Similarly, zero laxity execution cannot increase the amount of interference from a higher priority task with no carry-in job, given by Equation (5). This is an important observation. It means that when calculating interference from higher priority tasks, we do not need to know if they are zero laxity tasks.

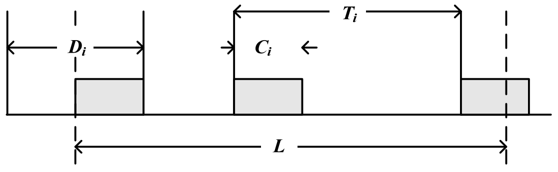

Figure 2

Under FPZL, each task τ_k is therefore schedulable without entering the zero laxity state if the following inequality holds:

$$D_k \ge C_k + \left\lfloor \frac{1}{m} \left(\begin{array}{c} \sum\limits_{\forall i \in hp(k)} I_i^{NC}(D_k) + \\ \sum\limits_{i \in Max(k,m-1)} I_i^{DIFF-D}(D_k) + \\ \sum\limits_{\forall j \in lpzl(k)} I_j^Z(D_k) \end{array} \right) \right\rfloor \qquad (19)$$

where $I_i^{NC}(D_k)$ is given by Equation (5), $I_i^{DIFF-D}(D_k)$ is given by Equation (8), $I_j^Z(D_k)$ is given by Equation (18), and *lpzl*(k) is the set of zero laxity tasks with lower priorities than k.

If the inequality in Equation (19) does not hold, then the task is a zero laxity task. Under FPZL, at most m tasks can be zero laxity tasks without a deadline being missed.

We note that the zero laxity status of each task is unknown until its schedulability is checked via Equation (19), hence task schedulability needs to be checked in priority order, lowest priority first.

Algorithm 1 presents the DA-LC schedulability test for FPZL. Note, for now we make the pessimistic assumption that a zero laxity task completes all of its execution in the zero laxity state, hence the line 'Compute Z_k^{UB}' can be assumed to set $Z_k^{UB} = C_k$. Section 5 describes the calculation of less pessimistic upper bounds on zero laxity execution.

The DA-LC schedulability test for FPZL is a polynomial time test requiring $O(n^2)$ operations (assuming that 'Compute Z_k^{UB}' takes linear time). Note that identifying the tasks with the *m*-1 largest values of $I_i^{DIFF-D}(D_k)$ can be achieved using a linear time selection algorithm[19].

We note that the DA-LC schedulability test for FPZL reduces to the DA-LC test for global FP scheduling for any taskset that the latter finds schedulable. Hence, the DA-LC test for FPZL dominates the DA-LC test for global FP scheduling.

```
1   countZL = 0
2   for (each priority level k, lowest first) {
3       Determine schedulability of $\tau_k$ according to
4       Equation (19)
5       if( $\tau_k$ is not schedulable) {
6           mark $\tau_k$ as a 'zero laxity' task
7           countZL = countZL + 1
8           Compute $Z_k^{UB}$
9       }
10  }
11  if(countZL > m)
12      return unschedulable
13  else
14      return schedulable
```

Algorithm 1: DA-LC schedulability test for FPZL

4.2. Response Time Analysis for FPZL

Building on the work of Bertogna and Cirinei [15] and Guan et al. [28] (Equation (14)), we now derive a sufficient schedulability test for FPZL which computes an upper bound R_k^{UB} on the response time of each task τ_k.

If task τ_k is schedulable under FPZL with a response time bounded by R_k^{UB}, then an upper bound on the interference in an interval of length R_k^{UB} due to a lower priority task τ_j executing in the zero laxity state can be obtained by substituting R_k^{UB} for the length of the interval in Equation (18).

We observe that again the maximum interference on task τ_k due to a lower priority zero laxity task τ_j is the same irrespective of whether τ_j is considered as having a carry-in job or not. Hence task τ_j contributes zero additional carry-in interference and so does not need to be included when determining the *m*-1 tasks that contribute the largest amounts of additional carry-in interference.

In the previous section, we showed that using deadline analysis the maximum interference on task τ_k from a higher priority task τ_i capable of entering the zero laxity state can be determined using Equation (1) assuming a carry-in job, and using Equation (5) assuming no carry-in job. Thus we showed that the maximum interference from a higher priority task is independent of whether or not that task is a zero laxity task.

We now consider the situation when response time analysis (Equation (10) and Equation (5)) is used to compute the interference due a higher priority zero laxity task. As the upper bound response time R_i^{UB} of a zero laxity task is equal to its deadline D_i, we find that Equation (10) reduces to Equation (1). Hence the maximum interference from a higher priority task is again independent of whether or not that task is a zero laxity task. An upper bound on the worst-case response time of a task τ_k that is schedulable under FPZL without entering the zero laxity state can therefore be found using the fixed point iteration given by Equation (20).

$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m} \left(\begin{array}{c} \sum_{\forall i \in hp(k)} I_i^{NC}(R_k^{UB}) + \\ \sum_{i \in Max(k,m-1)} I_i^{DIFF}(R_k^{UB}) + \\ \sum_{\forall j \in lpzl(k)} I_j^{Z}(R_k^{UB}) \end{array} \right) \right\rfloor \tag{20}$$

where $I_i^{NC}(R_k^{UB})$ is given by Equation (5), $I_i^{DIFF-R}(R_k^{UB})$ is given by Equation (14), and $I_j^{Z}(R_k^{UB})$ is given by Equation (18).

Iteration starts with $R_k^{UB} = C_k$, and continues until the value of R_k^{UB} converges in which case τ_k is schedulable, or until $R_k^{UB} > D_k$. If $R_k^{UB} > D_k$, then the task is a zero laxity task. Recall that under FPZL, at most *m* tasks may be zero laxity tasks without a deadline being missed.

Using Equation (20), we can construct a sufficient schedulability test for FPZL based on upper bound response times; however, this is not entirely straightforward. Equation (20) depends on the response times of higher priority tasks (via Equation (11)), and also on which lower priority tasks are zero laxity tasks. Thus it would appear that we cannot compute task schedulability in increasing or decreasing priority order. This problem can however be solved by backtracking as shown in Algorithm 2.

Algorithm 2 initially assumes that there are no zero laxity tasks and starts computing task response times in priority order, highest priority first (lines 6 and 7). Then, whenever a task τ_k is encountered where Equation (20) results in a value of $R_k^{UB} > D_k$, the task is marked as a zero laxity task and its upper bound response time is set to its deadline (lines 8 and 9). We note that provided that the taskset is schedulable under FPZL, then this is the correct upper bound response time, as the zero laxity rule will prevent the task from actually missing its deadline.

The discovery of a zero laxity task effectively invalidates the upper bound response times calculated for all higher priority tasks. These values could be too small, and therefore the process of upper bound response time calculation needs to be repeated (line 13). However, if more than *m* zero laxity tasks have been found, then even the zero laxity rule cannot prevent deadline misses and the taskset is deemed unschedulable. In this case, the algorithm can exit immediately (lines 15-17).

We note that lines 20-21 are not required when a simple fixed value of $Z_k^{UB} = C_k$ is used for the zero laxity execution time of task τ_k. However, when the computed value of Z_k^{UB} depends on the response times of higher priority tasks then this additional convergence check is required. This point is discussed in detail in Section 5.

```
1   countZL = 0
2   Initialize all $R_k^{UB} = C_k$ and $Z_k^{UB} = 0$
3   repeat = true
4   while (repeat) {
5       repeat = false
6       for (each priority level k, highest first) {
7           Determine $R_k^{UB}$ according to Eq. (20)
8           if( $R_k^{UB} > D_k$ ) {
9               $R_k^{UB} = D_k$
10              Compute $Z_k^{UB}$
11              if ( $\tau_k$ not marked as a ZL task) {
12                  mark $\tau_k$ as a ZL task
13                  repeat = true
14                  countZL = countZL + 1
15                  if(countZL > m) {
16                      repeat = false
17                      break (exit for loop)
18                  }
19              }
20          }
21          [if( $R_k^{UB}$ or $Z_k^{UB}$ differ from prev. values)
22              repeat = true]
23      }
24  }
25  if(countZL > m)
26      return unschedulable
27  else
28      return schedulable
```

Algorithm 2: RTA-LC schedulability test for FPZL

We note that the upper bound response time for a task τ_i is monotonically non-decreasing in the amount of zero laxity execution time of each of the tasks with lower priority than *i*. Hence, the calculation of R_i^{UB} can be made more efficient on subsequent iterations of the 'while' loop (line 4) by using as an initial value, the value of R_i^{UB} computed on the previous iteration.

The 'while' loop (lines 4-24) continues to iterate until either *m*+1 zero laxity tasks are found, in which case the taskset is unschedulable under FPZL, or there are *m* or fewer zero laxity tasks and the upper bound response times have been re-calculated since the final zero laxity task was found. In this case, the taskset is schedulable.

Under the assumption that 'Compute Z_k^{UB}' sets $Z_k^{UB} = C_k$, the RTA-LC schedulability test for FPZL requires at most $O(mn)$ response time calculations ((Equation (20)), each of which is pseudo-polynomial in complexity. By comparison, the RTA-LC test for global FP scheduling requires $O(n)$ such response time calculations.

We note that the RTA-LC schedulability test for FPZL reduces to the RTA-LC test for global FP scheduling for any taskset that the latter finds schedulable. Hence, the RTA-LC test for FPZL dominates the RTA-LC test for global FP scheduling. Further, the RTA-LC test for FPZL also dominates the DA-LC test for FPZL.

5. Bounding zero laxity execution time

So far, we have made the potentially pessimistic assumption that a task that can reach the zero laxity state does so without having started to execute. Hence, we used an upper bound on the zero laxity execution time of $Z_k^{UB} = C_k$. In this section, we derive a more effective upper bound and use this bound to improve the schedulability tests derived in Section 4. First, we introduce the concept of DC-Sustainability and prove that the schedulability tests for task τ_k given by Equations (19) and (20) are DC-Sustainable.

5.1. DC-Sustainability

A schedulability test for task τ_k is referred to as *DC-Sustainable* if it is sustainable [9] with respect to simultaneous and equal changes in both the execution time and the deadline of the task. Below we give a formal definition of DC-Sustainability.

Definition: A schedulability test *S* for a task τ_k is *DC-Sustainable* if the following two conditions hold:

Condition 1: If task τ_k is deemed schedulable by test *S* with some paired deadline and execution time values $D'_k = D_k - v$, $C'_k = C_k - v$ where $0 \le v \le C_k$ then test *S* is guaranteed to deem task τ_k schedulable for all deadline and execution time pairs $D'_k = D_k - w$, $C'_k = C_k - w$ where $v \le w \le C_k$.

Condition 2: If task τ_k is deemed unschedulable by test *S* with some paired deadline and execution time values $D'_k = D_k - v$, $C'_k = C_k - v$ where $0 \le v \le C_k$ then test *S* is guaranteed to deem task τ_k unschedulable for all deadline and execution time pairs $D'_k = D_k - w$, $C'_k = C_k - w$ where $0 \le w \le v$.

Theorem 2: Equation (19) is a DC-Sustainable schedulability test for task τ_k.

Proof: We can re-write Equation (19) as follows:

$$D'_k - C'_k \ge + \left\lfloor \frac{1}{m} \left(\begin{array}{l} \sum\limits_{\forall i \in hp(k)} I_i^{NC}(D'_k) + \\ \sum\limits_{i \in Max(k,m-1)} I_i^{DIFF-D}(D'_k) + \\ \sum\limits_{\forall j \in lpzl(k)} I_j^{Z}(D'_k) \end{array} \right) \right\rfloor \tag{21}$$

Consider the behaviour of Equation (21) for paired deadline and execution time values $D'_k = D_k - w$, $C'_k = C_k - w$ as *w* takes different values in the range $0 \le w \le C_k$. The RHS of Equation (21) gives an upper bound on the interference from higher priority tasks and lower priority tasks executing in the zero laxity state in an interval of length $D'_k = D_k - w$. By inspecting the component Equations (1), (2), (3), (5), (6), (7), (8), (16), (17), and (18) it can be seen that this interference is monotonically non-decreasing with respect to the length of the interval D'_k. We must however also consider the dependence of component Equations (5) and (18) on C'_k, which also varies with *w*. C'_k appears in the second term in the *min*( ) function of each of these equations in the expression $D'_k - C'_k + 1$. This expression is unchanged by varying *w*. The RHS of Equations (21) is therefore monotonically non-increasing with respect to increasing values of *w*.

In the case of Condition 1, as the LHS of Equation (21) is unchanged and the RHS is monotonically non-increasing for increasing values of w: $0 \le w \le C_k$ then it follows that, given that Equation (21) holds for w=v, it must also hold for all values of w: $v \le w \le C_k$.

In the case of Condition 2 as the LHS of Equation (21) is unchanged and the RHS is monotonically non-decreasing for decreasing values of w: $0 \le w \le C_k$ then it follows that, given that Equation (21) does not hold for w=v, then it cannot hold for any value of w: $0 \le w \le v$ □

We now prove that Equation (20) is also a DC-Sustainable schedulability test for task τ_k. Below, we re-write Equation (20), using the variable q to indicate the fixed point iteration.

$$R_k^{q+1} \leftarrow C'_k + \left\lfloor \frac{1}{m} \left(\begin{array}{c} \sum_{\forall i \in hp(k)} I_i^{NC}(R_k^q) + \\ \sum_{i \in Max(k,m-1)} I_i^{DIFF}(R_k^q) + \\ \sum_{\forall j \in lpzl(k)} I_j^{Z}(R_k^q) \end{array} \right) \right\rfloor \quad (22)$$

Recall that iteration begins with $R_k^0 = C'_k$ (the execution time of task τ_k), and ends when either $R_k^{q+1} = R_k^q$ or when $R_k^{q+1} > D'_k$, in which case task τ_k is unschedulable.

Let $R_k^{UB}(D,C)$ be the response time upper bound given by Equation (22) for task τ_k (D, C) with deadline D and execution time C. Similarly, let $R_k^{UB}(D+x, C+x)$ be the response time upper bound given by Equation (22) for task $\tau_k(D+x, C+x)$ with deadline $D+x$ and execution time $C+x$.

Lemma 1: If τ_k (D, C) is schedulable according to Equation (22) then $R_k^{UB}(C+x) \ge R_k^{UB}(C) + x$. Further, if τ_k (D, C) is not schedulable according to Equation (22) then neither is $\tau_k(D+x, C+x)$.

Proof: Let $R_k^q(C)$ be the value computed by the qth iteration of Equation (22) for task τ_k (D, C). Similarly, let $R_k^q(C+x)$ be the value computed by the qth iteration of Equation (22) for task $\tau_k(D+x, C+x)$.

We prove the Lemma by induction, showing that on every iteration q until convergence or the deadline of τ_k (D, C) is exceeded, $R_k^q(C+x) \ge R_k^q(C) + x$.

Initial condition: in each case iteration starts with an initial value corresponding to the execution time of τ_k, hence $R_k^0(C) = C$ and $R_k^0(C+x) = C+x$, so $R_k^0(C+x) \ge R_k^0(C) + x$.

Inductive step: assume that $R_k^q(C+x) \ge R_k^q(C) + x$, and consider the values computed for $R_k^{q+1}(C+x)$ and $R_k^{q+1}(C)$ on iteration q+1. The floor function (second term on the RHS of Equation (22)) contains three summation terms; together, these terms give an upper bound on the interference from higher priority tasks and lower priority tasks executing in the zero laxity state in an interval of length R_k^q. Inspection of the component Equations ((5), (6), (7), (10), (11), (12), (14), (16), (17), and (18)) shows that this interference term is no smaller for input values $R_k^q(C+x) \ge R_k^q(C) + x$, and $C'_k = C + x$ (the latter is used in Equations (10) and (18)) than it is for input values $R_k^q(C)$ and $C'_k = C$, hence once the value of C'_k is added (first term on the RHS of Equation (22)), we have $R_k^{q+1}(C+x) \ge R_k^{q+1}(C) + x$.

We note that if the fixed point iteration for τ_k (D, C) converges on $R_k^{UB}(D,C) = R_k^{q+1}(C)$, then the smallest possible value of $R_k^{UB}(D+x, C+x)$ is $R_k^{q+1}(C) + x$. Further, if τ_k (D, C) is unschedulable, then it follows that $R_k^{q+1}(C) > D$ which implies that $R_k^{q+1}(C+x) > D + x$ and therefore $\tau_k(D+x, C+x)$ must also be unschedulable □

Theorem 3: Equation (22), and hence Equation (20) is a DC-Sustainable schedulability test.

Proof: We can choose an execution time of $C'_k = 0$ and a deadline of $D'_k = D_k - C_k$ for task τ_k. With these parameters, τ_k is deemed schedulable by Equation (22). We then consider all possible deadline and execution time pairs $D'_k = D_k - w$, $C'_k = C_k - w$ for w from 1 to C_k (recall that execution times are represented by non-negative integers). Let v be the smallest value of w, if any, for which τ_k is unschedulable. Lemma 1 tells us that for all larger values of w, τ_k will also be unschedulable. Proof that Conditions 1 and 2 in the definition of DC-Sustainability hold follows directly from the observation that task schedulability is monotonically decreasing with respect to increasing values of w □

5.2. Zero laxity execution time

We now show how a bound on the zero laxity execution time of each zero laxity task can be derived. Let us assume that we are using the DA-LC schedulability test (Algorithm 1) or the RTA-LC schedulability test (Algorithm 2) for FPZL, and that task τ_k has been identified as a zero laxity task by Equation (19) or Equation (20). We know that task τ_k cannot be guaranteed to complete all of its execution within its deadline, without entering the zero laxity state. However, if we can show that τ_k is guaranteed to complete $C'_k = C_k - v$ units of execution time by an effective deadline of $D'_k = D_k - v$, without entering the zero laxity state, then that proves that the task can execute for at most v units of time in the zero laxity state. (Note we only need consider a task as entering the zero laxity state if its remaining execution time is equal to the elapsed time to its deadline *and* it is not one of the m highest priority ready tasks).

Due to the DC-Sustainability of the single task schedulability tests given by Equations (19) and (20), each of these equations can be used as the basis of a binary search to determine the smallest value of v $(0 \le v \le C_k)$ such that task τ_k is guaranteed to complete $C'_k = C_k - v$ units of execution time by a deadline $D'_k = D_k - v$, thus computing an upper bound $Z_k^{UB} = v$ on the amount of time that a job of task τ_k can spend executing in the zero laxity state. The initial minimum value of v for the search is $v = 0$ which is known to result in un-schedulability, as τ_k is a zero laxity task, while the initial maximum value is $v = C_k$ which is deemed to result in schedulability, as it is equivalent to τ_k having zero execution time.

In the DA-LC test, a binary search based on Equation (19) can be used to 'Compute Z_k^{UB}' (line 8 of Algorithm 1), for each zero laxity task, improving the effectiveness of the test. Similarly, in the RTA-LC test, a binary search based on Equation (20) can be used to 'Compute Z_k^{UB}' (line 10 of Algorithm 2) for each zero laxity task. However, in this case, a further convergence check (lines 20-21) is required as the zero laxity execution times computed by the binary searches are dependent on the response times of higher priority tasks, and vice-versa. We note that Algorithm 2 will either find more than m zero laxity tasks or converge on unchanging values for the response times and zero laxity execution times. Such convergence is guaranteed because the response times of higher priority tasks are monotonically non-decreasing with respect to increases in the zero laxity execution time of lower priority tasks, and similarly, the zero laxity execution times of lower priority tasks computed by binary search are monotonically non-decreasing with respect to increases in the response times of higher priority tasks.

6. Priority assignment

In this section, we briefly discuss priority assignment for FPZL. Davis and Burns [24] showed that priority assignment is a key factor in global FP scheduling. As FPZL is a hybrid of global FP scheduling with the addition of the zero laxity rule, we expect priority assignment to also be important in FPZL scheduling.

The DA-LC and RTA-LC schedulability tests for FPZL given by Algorithm 1 and Algorithm 2 respectively are independent of the priority ordering used. Hence they are compatible with heuristic priority assignment policies such as Deadline Monotonic Priority Ordering (DMPO) or DkC [24]. When there are no zero laxity tasks, FPZL reduces to global FP scheduling. In this case, Audsley's Optimal Priority Assignment (OPA) algorithm [2], [3] provides the optimal priority assignment to use in conjunction with the DA-LC test for FPZL. However, when the OPA algorithm finds that there are no tasks that are schedulable at a particular priority level without recourse to the zero laxity rule, then the following question arises: Which task should be assigned to that priority level? For the purposes of the empirical evaluation in Section 7, we answered this question via a simple heuristic. We computed the zero laxity execution time for each unassigned task using a binary search, and assigned the task with the smallest proportion of its execution time in the zero laxity state. The idea being that this is the task that would require the smallest percentage reduction in its execution time to be schedulable at that priority without recourse to the zero laxity rule.

Full investigation of priority assignment policies for FPZL is an interesting area for future research.

7. Empirical investigation

In this section, we present the results of an empirical investigation, examining the effectiveness of the schedulability tests for FPZL. We also conducted scheduling simulations which form necessary but not sufficient schedulability tests, thus providing upper bounds on the potential performance of the various scheduling algorithms.

7.1. Taskset parameter generation

The taskset parameters used in our experiments were randomly generated as follows:

- o Task utilisations were generated using the UUnifast-Discard algorithm [24], giving an unbiased distribution of task utilisations. A discard limit of 1000 was used, but not needed.
- o Task periods were generated according to a log-uniform distribution with a factor of 1000 difference between the minimum and maximum possible task period. This represents a spread of task periods from 1ms to 1 second, as found in most hard real-time applications. The log-uniform distribution was used as it generates an equal number of tasks in each time band (e.g. 1-10ms, 10-100ms etc.), thus providing reasonable correspondence with real systems.
- o Task execution times were set based on the utilisation and period selected: $C_i = U_i T_i$.
- o To generate constrained-deadline tasksets, task deadlines were assigned according to a uniform random distribution, in the range $[C_i, T_i]$. For implicit-deadline tasksets, deadlines were set equal to periods.

In each experiment, the taskset utilisation (x-axis value) was varied from 0.025 to 0.975 times the number of processors in steps of 0.025. For each utilisation value, 1000 valid tasksets were generated and the schedulability of those tasksets determined using the various schedulability tests for different scheduling algorithms. The graphs plot the percentage of tasksets generated that were deemed schedulable in each case. Note the lines on all of the graphs appear in the order given in the legend. (The graphs are best viewed online in colour).

7.2. Scheduling simulation

We used a simulation of global FP, FPZL, global EDF and EDZL scheduling to provide an upper bound on the potential performance of each scheduling algorithm, and hence to evaluate the quality of the schedulability tests. Our simulations ran for an interval of time equal to ten times the longest period of any task in the taskset. Each simulation started with synchronous release of the first job of each task, with subsequent jobs released as early as possible. Each job executed for its worst-case execution time. The simulation deemed a taskset schedulable by a given algorithm if it did not find a deadline miss during the time interval simulated, or any unavoidable deadline miss for any job that had execution time remaining at the end of the interval. Thus the simulation provides a necessary but not sufficient schedulability test. Any taskset failing the simulation, with a deadline miss, is guaranteed to be unschedulable, while tasksets that pass the simulation may or may not be schedulable. We note that in the case of constrained-deadline sporadic tasksets, to the best of our knowledge, no tractable exact tests exist for any of the algorithms studied. Thus upper bounds on performance derived via simulation are one of the few ways in which the performance potential of each algorithm can be explored.

7.3. Schedulability test effectiveness

We investigated the performance of the FPZL DA-LC schedulability test using Audsley's OPA algorithm [2], [3] to assign priorities, and compared its performance to that of that of the equivalent tests for global FP scheduling, and to the most effective schedulability tests currently known for global EDF [17] (the "EDF-RTA" test) and EDZL scheduling [7] (the "EDZL-I test"). Also shown on the graphs are results for the necessary infeasibility test of Baker and Cirinei [6] (labelled "LOAD*"). This line gives the total number of tasksets at each utilisation level that we cannot be certain are infeasible (i.e. unschedulable by any algorithm). Further, the narrow lines on the graphs indicate an upper bound on the performance of each algorithm found via simulation. In the case of global FP and FPZL scheduling, these upper bounds assume Deadline minus Computation time Monotonic Priority Ordering (DCMPO) [24], which was found in the simulation studies to be significantly more effective than Deadline Monotonic Priority Ordering (DMPO).

Figures 3 to 6 below are for constrained-deadline tasksets. From these graphs, we can see that the EDF-RTA test for global EDF scheduling and the DA-LC test for global FP scheduling using DMPO have the lowest performance, with approximately 50% of the generated tasksets schedulable at a utilisation of 2.7 (=0.34*m*) and 2.8 (=0.35*m*) respectively, in the 8 processor case. The EDZL-I test performs significantly better with 50% of the tasksets schedulable at a utilisation of approx. 3.4 (=0.43*m*). Using optimal priority assignment significantly improves the performance of global FP scheduling, with 50% of the tasksets schedulable at a utilisation of approximately 4.7 (=0.59*m*) according to the DA-LC test. Finally, the DA-LC test for FPZL, using Audsley's OPA algorithm and a binary search to bound zero laxity execution time (marked FPZL-LZ on the graph) has the highest performance, with 50% of tasksets deemed schedulable at a utilisation of approx. 4.9 (=0.61*m*); a modest improvement over global FP scheduling.

Our simulation results show that both global EDF and global FP scheduling with DMPO have relatively poor performance potential. This is because these algorithms typically favour executing tasks with short deadlines first. This has the effect of reducing the amount of available concurrency, in terms of the number of ready tasks, which makes the remaining tasks more difficult to schedule. By contrast, using DCMPO greatly improves the performance potential of global FP scheduling, particularly when there are a large number of processors and tasks. The simulation results show that EDZL and FPZL with DCMPO priority ordering have similar performance potential, which as the number of processors and tasks increases becomes close to the upper bound given by the LOAD* infeasibility test.

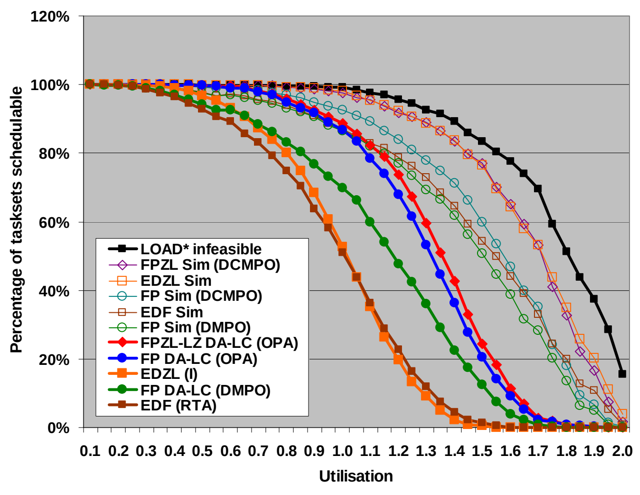

Figure 3: (2 processors, 10 tasks, D≤T)

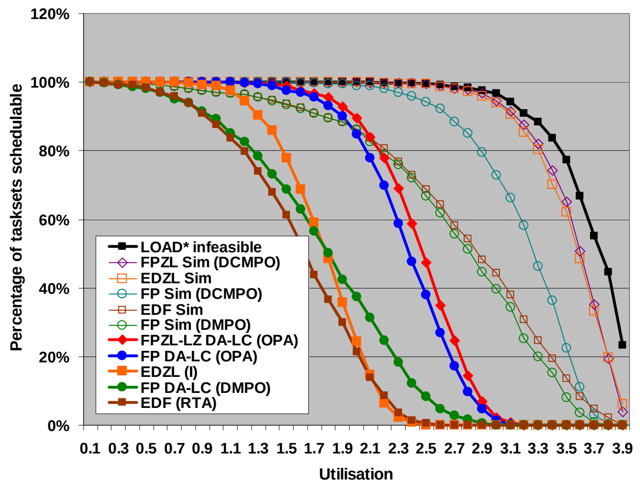

Figure 4: (4 processors, 20 tasks, D≤T)

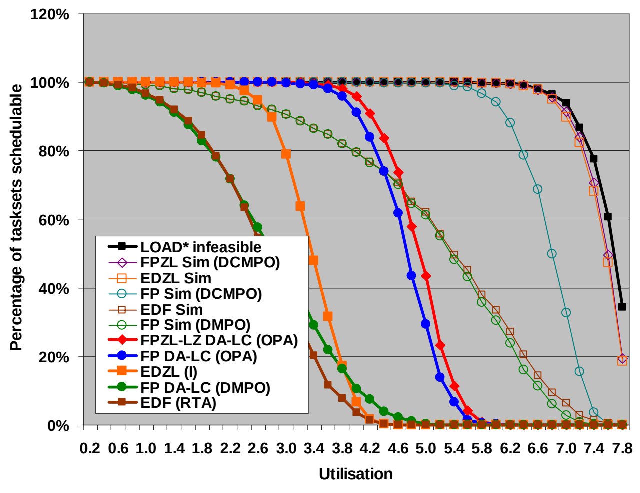

Figure 5: (8 processors, 40 tasks, D≤T)

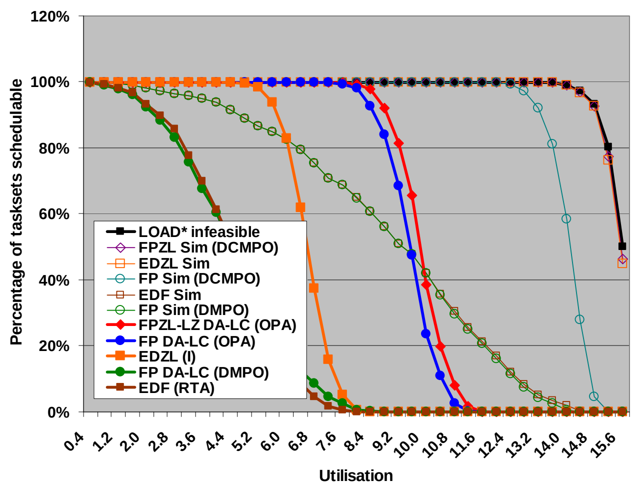

Figure 6: (16 processors, 80 tasks, D≤T)

Figures 7 to 10 below show the results of the same experiments, repeated for implicit-deadline tasksets. These graphs show that the performance of the schedulability tests for FPZL significantly exceed that of the best known tests for global FP, global EDF and EDZL, with an increased gap between FPZL and global FP scheduling using OPA, compared to the constrained deadline case. For example, in the 8 processor case, approximately 50% of the generated tasksets were schedulable at a utilisation of 6.1 (=0.76*m*) using FPZL (OPA), compared to 5.8 (=0.725*m*) for global FP scheduling using OPA, and 5 (=0.63) for EDZL(I). This increase in the relative performance of FPZL is mainly due to the calculation of a less pessimistic bound on the amount of zero-laxity execution time having an increased effect compared to the constrained-deadline case. Further, the simulation results show that the performance potential of EDZL and FPZL with DCMPO is very similar, with both algorithms potentially able to schedule nearly all of the tasksets generated.

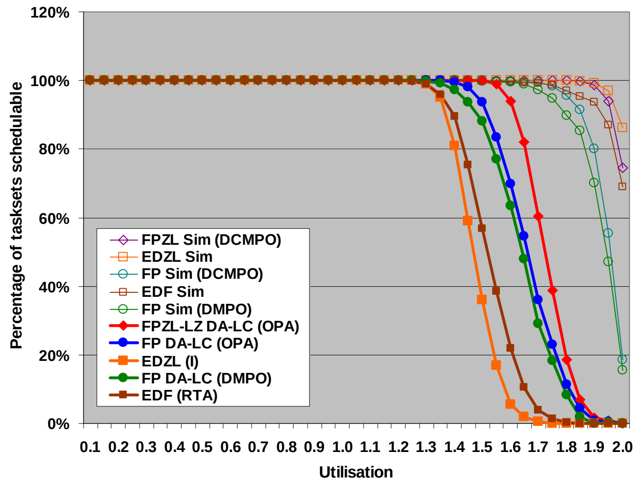

Figure 7: (2 processors, 10 tasks, D=T)

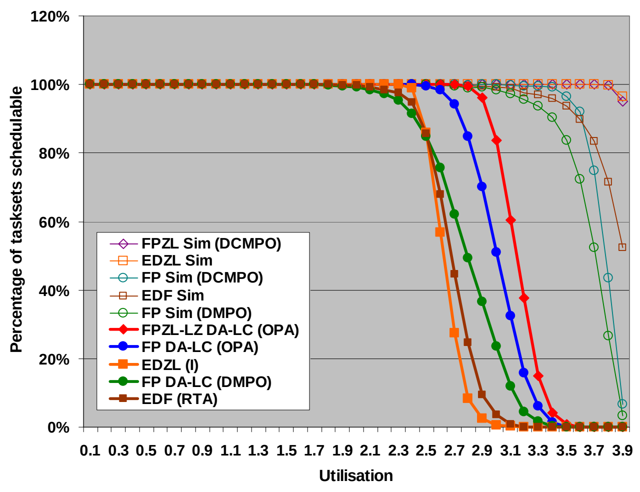

Figure 8: (4 processors, 20 tasks, D=T)

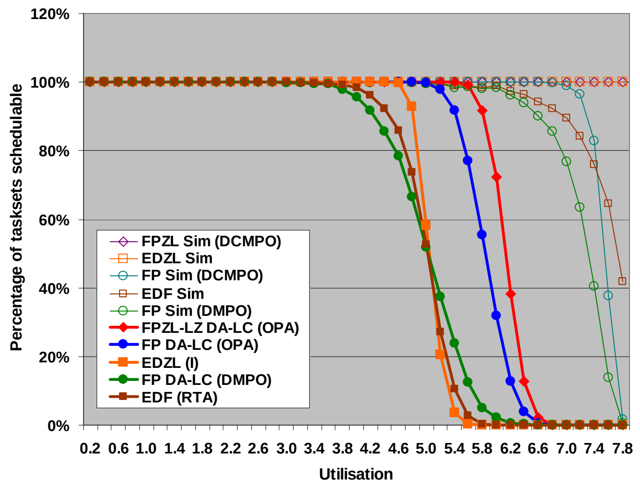

Figure 9: (8 processors, 40 tasks, D=T)

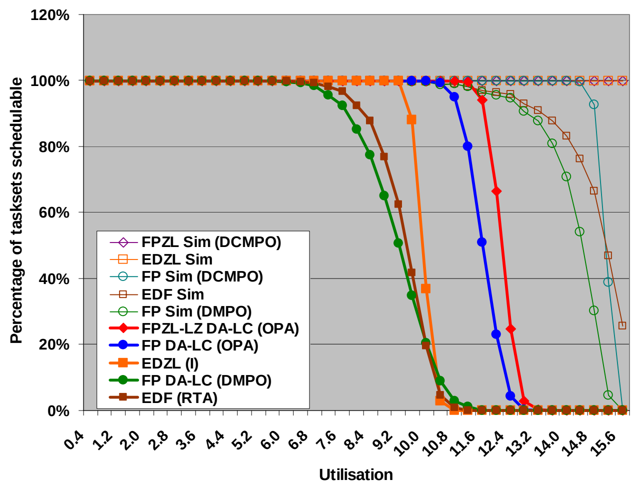

Figure 10: (16 processors, 80 tasks, D=T)

For implicit-deadline tasksets, we used our experimental results to obtain approximate values for the *Optimality Degree* (OD) [18] of each scheduling algorithm / schedulability test examined, over a domain corresponding to the tasksets generated in our experiments.

The Optimality Degree of a scheduling algorithm *A* combined with a schedulability test *S* is defined with respect to a domain of tasksets. It is given by the number of tasksets in the domain that are schedulable using algorithm *A* according to schedulability test *S*, divided by the number of feasible tasksets in the domain. Hence an optimal algorithm supported by an exact schedulability test has OD = 1 for any domain.

For sporadic tasksets with implicit-deadlines, the utilisation bound for LRE-TL [27] is 100%, hence all of the implicit-deadline tasksets generated in our experiments are feasible (as their utilisation does not exceed *m*). For each of the algorithms / schedulability tests examined, an approximate value for the Optimality Degree can therefore be obtained by simply counting the total number of schedulable tasksets over the full range of utilisation values, and dividing this number by the total number of tasksets generated. The Optimality Degree of each algorithm / schedulability test is given in Table 1 below, expressed as a percentage.

Table 1: Approximate Optimality Degree

	#Processors			
Algorithm / test	**2**	**4**	**8**	**16**
FPZL LZ DA-LC (OPA)	84.7%	79.4%	77.4%	76.6%
FP DA-LC (OPA)	81.4%	75.7%	73.6%	73.2%
FP DA-LC (DMPO)	80.1%	70.0%	62.8%	57.2%
EDZL(I)	71.7%	66.2%	63.5%	62.3%
EDF (RTA)	74.2%	67.4%	62.5%	58.5%

Table 1 shows that the Optimality Degree for FPZL scheduling using the polynomial time DA-LC schedulability test derived in this paper, (with OPA priority assignment and zero laxity execution time calculation) is 3-4% better than for global FP scheduling using OPA and an equivalent schedulability test, and 13% better than for EDZL, assuming the iterative schedulability test given in [7].

We repeated our experiments for larger numbers of tasks per processor (20) and for a smaller range of task periods (with a factor of ten difference between the minimum and maximum possible period). In each case, although the data points changed, the relationships between the effectiveness of the different methods and the conclusions that can be drawn from them remained essentially the same. As the number of tasks per processor increased, we observed the following minor changes:

- The effectiveness of the schedulability tests for FPZL and global FP scheduling increased, while the effectiveness of the schedulability tests for global EDF and EDZL declined.
- The potential performance of EDZL exceeded that of FPZL (DCMPO) by a small margin.

7.4. Effectiveness of zero laxity bounds

In this section, we examine the relative improvements in the FPZL schedulability tests obtained by (i) computing a more effective bound on the zero laxity execution time, (ii) using the pseudo-polynomial time RTA-LC tests rather than the polynomial time DA-LC tests.

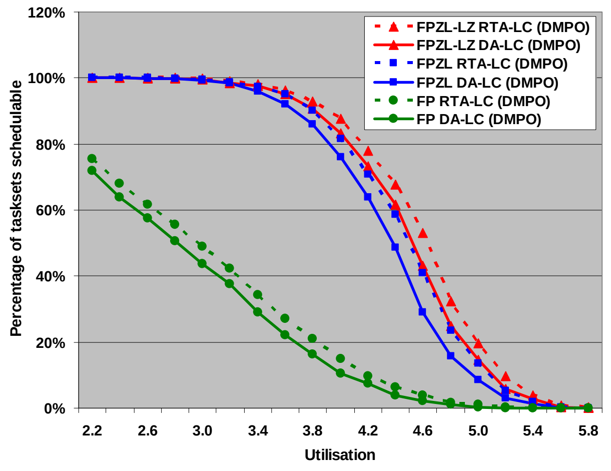

Figure 11: (8 processors, 40 tasks, D≤T)

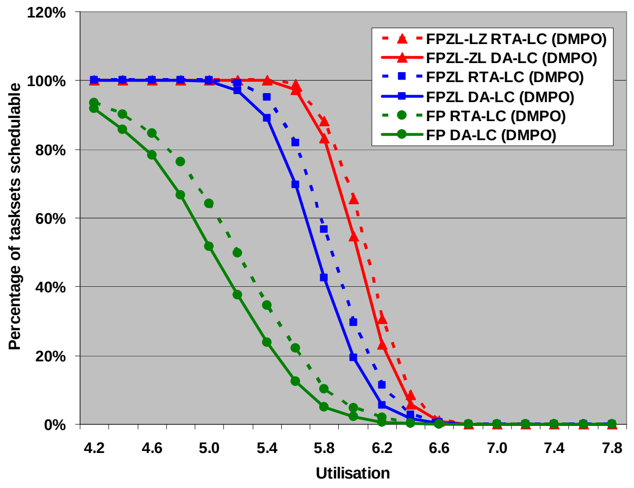

Figure 12: (8 processors, 40 tasks, D=T)

Figure 11 compares the performance of the DA-LC and RTA-LC schedulability tests for global FP and FPZL scheduling, using DMPO, for systems with 40 constrained-deadline tasks executing on 8 processors. There are 3 pairs of lines on the graph. Each pair illustrates the performance of the RTA-LC test (dashed line) compared to that of the DA-LC test (solid line). The lowest pair of lines are for global FP scheduling, the middle pair of lines are for FPZL with the upper bound zero laxity execution time Z_k^{UB} given by C_k, and the top pair of lines are for FPZL using a binary search to compute Z_k^{UB}. As expected the RTA-LC tests dominate the equivalent DA-LC tests. Further, the FPZL tests which use a binary search to provide a less pessimistic bound on the amount of zero laxity execution time dominate the equivalent tests that assume that all of a zero laxity task's execution is in the zero laxity state. We note that the improvement in performance achieved by computing a less pessimistic bound on zero laxity execution time is broadly similar to the improvement obtained by using the pseudo-polynomial time RTA-LC test as compared to using the polynomial-time DA-LC test. Figure 11 also shows that the performance of the FPZL schedulability tests greatly exceeds that of the equivalent tests for global FP scheduling when DMPO is used.

Figure 12 shows the results of the same experiment repeated for implicit-deadline tasksets. (Note the differing x-axis interval on the two graphs). Compared to the constrained-deadline case, the improvement in FPZL schedulability test performance obtained by calculating a less pessimistic bound on the amount of zero-laxity execution time is significantly enhanced. This improvement is represented by the gap between the two highest solid lines (DA-LC test) and between the two highest dashed lines (RTA-LC test). The relative performance of the schedulability tests for global FP scheduling with DMPO is also improved, although the absolute performance is still significantly worse than that of comparable tests for FPZL.

8. Conclusions and future work

The motivation for our work was the desire to improve upon the current state-of-the-art in terms of practical techniques that enable the efficient use of processing capacity in hard real-time systems based on multiprocessors.

The intuition behind our work was that dynamic priority scheduling has the potential to schedule many more tasksets than fixed task or fixed job priority algorithms, and yet this theoretical advantage has to be tempered by the need to avoid prohibitively large overheads due to a high number of pre-emptions. This led us to consider minimally dynamic scheduling algorithms which permit each job to change priority at most once during its execution. One such algorithm is EDZL. We applied the zero laxity rule from EDZL to global FP scheduling, forming the FPZL scheduling algorithm. The number of context switches with FPZL is at most two per zero laxity task, and one per ordinary task. As there are at most *m* zero laxity tasks, the increase in overheads compared to global FP scheduling is tightly bounded.

The key contributions of this paper are as follows:

- The introduction of the FPZL scheduling algorithm.
- The derivation of effective polynomial time and pseudo-polynomial time sufficient schedulability tests for FPZL, based on similar tests for global FP scheduling.
- Improvements to these tests, bounding the amount of execution that may take place in the zero laxity state.

The main conclusions that can be drawn from our empirical investigations are as follows:

- The zero laxity rule employed by FPZL appears to have a large impact on taskset schedulability, compared to the performance of global FP scheduling, as shown by the simulation results. The performance potential of FPZL using DCMPO was found to be broadly similar to that of EDZL, and significantly better than that of global FP or global EDF scheduling.
- Using Audsley's OPA algorithm to assign task priorities, the polynomial time schedulability test for FPZL results in a modest improvement over the equivalent test for global FP scheduling in the case of constrained-deadline tasksets, with an increased improvement for implicit-deadline tasksets.
- The schedulability tests for FPZL derived in this paper, and the best known schedulability tests for global FP scheduling, appear to significantly outperform the best known tests for global EDF and EDZL. Even so, there remains a large gap between the sufficient schedulability tests for FPZL and what might be possible as shown by the simulation results.

Given the similarities between FPZL and EDZL, it is interesting to consider why the schedulability tests for FPZL significantly outperform those for EDZL. All of these schedulability tests are sufficient, and so suffer from a degree of pessimism in terms of the computed interference. The advantage that the schedulability tests for FPZL have over those for EDZL is that this pessimism is restricted to tasks with higher priorities and lower priority zero-laxity tasks. With the schedulability tests for EDZL (and EDF), there is pessimism attributable to the calculation of interference from *all* other tasks. Further, the techniques derived in this paper, reduce the amount of interference considered due to tasks executing in the zero-laxity state, by bounding the amount of execution that takes place in that state. Nevertheless, the tests for FPZL have an additional element of pessimism compared to similar tests for global FP scheduling due to the inclusion of zero laxity tasks in the interference term. This may account for the fact that the difference in performance between the schedulability tests for FPZL and global FP scheduling is not as large as the difference in the potential performance of the two algorithms as shown by simulation.

In future, we intend to investigate priority assignment policies for FPZL, including how task priorities should be assigned when it is inevitable that there will be some zero laxity tasks. We also intend to look at variants of FPZL based on the idea of critical-laxity and EDCL [31].

8.1. Acknowledgements

This work was funded by the EPSRC Tempo project (EP/G055548/1) and the Artist Design Network of Excellence.

References

[1] B. Andersson, J. Jonsson, "Some insights on fixed-priority pre-emptive non-partitioned multiprocessor scheduling". In Proc. RTSS – Work-in-Progress Session, Nov. 2000.

[2] N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, Dec. 1991.

[3] N.C. Audsley, "On priority assignment in fixed priority scheduling", Information Processing Letters, 79(1): 39-44, May 2001.

[4] T.P. Baker. "Multiprocessor EDF and deadline monotonic schedulability analysis". In Proc. RTSS, pp. 120–129, 2003.

[5] T.P. Baker. "An analysis of fixed-priority scheduling on a multiprocessor". Real Time Systems, 32(1-2), 49-71, 2006.

[6] T.P. Baker, M. Cirinei, "A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks", In proc. Work-In-Progress (WIP) session of RTSS 2006.

[7] T.P.Baker, M. Cirinei, M. Bertogna, "EDZL scheduling analysis". Real-Time Systems. 40:3, 264-289, 2008

[8] T.P. Baker, S.K. Baruah. "Sustainable multiprocessor scheduling of sporadic task systems". In Proc. ECRTS, pp. 141-150, 2009.

[9] S.K. Baruah., A. Burns, "Sustainable Scheduling Analysis". In Proc. RTSS, pp. 159-168, 2006.

[10] S.K. Baruah, "Techniques for Multiprocessor Global Schedulability Analysis". In Proc. RTSS, pp. 119-128, 2007.

[11] S.K. Baruah, N. Fisher. "Global Fixed-Priority Scheduling of Arbitrary-Deadline Sporadic ..." In Proc. of the 9th Int'l Conference on Distributed Computing and Networking, pp. 215-226, Jan 2008.

[12] S.K. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, "Implementation of a speedup-optimal global EDF schedulability test", In Proc. ECRTS, pp. 259-268, 2009.

[13] S.K. Baruah, T.P. Baker, "An analysis of global EDF schedulability for arbitrary sporadic task systems. Real-Time Systems ECRTS special issue, 43(1): 3-24, Sept. 2009.

[14] M. Bertogna, M. Cirinei, G. Lipari, "New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors". In Proc. 9th International Conf. on Principles of Distributed Systems, pp. 306-321, Dec. 2005.

[15] M. Bertogna, M. Cirinei, "Response Time Analysis for global scheduled symmetric multiprocessor platforms". In Proc. RTSS, pp. 149-158, 2007.

[16] M. Bertogna, "Real-Time Scheduling for Multiprocessor Platforms". PhD Thesis, Scuola Superiore Sant'Anna, Pisa, 2007.

[17] M. Bertogna, M. Cirinei, G. Lipari. "Schedulability analysis of global scheduling algorithms on multiprocessor platforms". IEEE Transactions on parallel and distributed systems, 20(4): 553-566. April 2009.

[18] E. Bini and G.C. Buttazzo. "Measuring the Performance of Schedulability tests". Real-Time Systems, 30(1–2):129–154, May 2005.

[19] M. Blum, R.W. Floyd, V. Pratt, R. Rivest and R. Tarjan, "Time bounds for selection," Journal of Computer and System Sciences. 7 (1973) pp. 448-461.

[20] Y-H Chao, S-S Lin, K-J Lin, "Schedulability issues for EDZL scheduling on real-time multiprocessor systems", Information Processing Letters, Volume 107, Issue 5, pp. 158-164, 16 August 2008

[21] M. Cirinei, T. P. Baker. "EDZL scheduling analysis". In Proc. ECRTS, pp. 9–18, 2007.

[22] H. Cho, B. Ravindran, E.D. Jensen, "An Optimal Real-Time Scheduling Algorithm for Multiprocessors". In Proc. RTSS pp. 1001-110, 2006.

[23] S. Cho, S-K. Lee, A. Han, K-J Lin, "Efficient real-time scheduling algorithms for multiprocessor systems". IEICE Transactions on Communications Vol. E85-B No. 12, pp.2859–2867, 2002.

[24] R.I. Davis, A. Burns, "Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems". In Proc. RTSS, pp. 398-409, 2009.

[25] R.I. Davis, A. Burns, "A Survey of Hard Real-Time Scheduling Algorithms and Schedulability Analysis Techniques for Multiprocessor Systems", Technical Report YCS-2009-443, Dept. of Computer Science, University of York, 2009.

[26] N. Fisher, S.K. Baruah. "Global Static-Priority Scheduling of Sporadic Task Systems on Multiprocessor Platforms." In Proc. IASTED International Conference on Parallel and Distributed Computing and Systems. Nov. 2006.

[27] S. Funk, V. Nadadur, "LRE-TL: An Optimal Multiprocessor Algorithm for Sporadic Task Sets". In Proc. RTNS, pp. 159-168, 2009.

[28] N. Guan, M. Stigge, W.Yi, G. Yu, "New Response Time Bounds for Fixed Priority Multiprocessor Scheduling". In Proc. RTSS, pp 387-397 2009.

[29] R. Ha, J. W.-S. Liu, "Validating timing constraints in multiprocessor and distributed real-time systems". In Proc. of the International conference on Distributed Computing Systems, pp. 162–171, 1994.

[30] S.K. Lee, "On-line multiprocessor scheduling algorithms for real-time tasks", In Proc. IEEE Region 10's Ninth Annual International Conference, pp. 607–611, 1994.

[31] S. Kato, N. Yamasaki, "Global EDF-based scheduling with efficient priority promotion". In Proc. of RTCSA pp. 197–206, 2008.

[32] Piao X, Han S, Kim H, Park M, Cho Y, Cho S "Predictability of earliest deadline zero laxity algorithm for multiprocessor real time systems". In: Proc. of the 9th IEEE international symposium on object and component-oriented real-time distributed computing, Gjeongju, Korea, 2006.

[33] M. Park, S. Han, H. Kim, S. Cho, Y. Cho, "Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor". IEICE Transactions on Information Systems Vol. E88-D No. 3, pp. 658–661, 2005.