IPP HURRAY!

www.hurray.isep.ipp.pt

Technical Report

LLF Schedulability Analysis on Multiprocessor Platforms

Jinkyu Lee

Arvind Easwaran

Insik Shin

HURRAY-TR-100902

Version:

Date: 09-09-2010

LLF Schedulability Analysis on Multiprocessor Platforms

Jinkyu Lee, Arvind Easwaran, Insik Shin

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

LLF (Least Laxity First) scheduling, which assigns a higher priority to a task with smaller laxity, has been known as an optimal preemptive scheduling algorithm on a single processor platform. However, its characteristics uponmultiprocessor platforms have been little studied until now. Orthogonally, it has remained open how to efficiently schedule general task systems, including constrained deadline task systems, upon multiprocessors. Recent studies have introduced zero laxity (ZL) policy, which assigns a higher priority to a task with zero laxity, as a promising scheduling approach for such systems (e.g., EDZL). Towards understanding the importance of laxity in multiprocessor scheduling, this paper investigates the characteristics of ZL policy and presents the first ZL schedulability test for any work-conserving schedulingalgorithm that employs this policy. It then investigates the characteristics of LLF scheduling, which also employs the ZL policy, and derives the first LLF-specific schedulability test on multiprocessors. It is shown that the proposed LLF test dominates the ZL test as well as the state-of-art EDZL test.

LLF Schedulability Analysis on Multiprocessor Platforms

Jinkyu Lee*, Arvind Easwaran† and Insik Shin*‡

*Dept. of Computer Science, KAIST, South Korea
†Cister Research Unit, Polytechnic Institute of Porto, Portugal
jinkyu@cps.kaist.ac.kr; aen@isep.ipp.pt; insik.shin@cs.kaist.ac.kr

***Abstract*—LLF (Least Laxity First) scheduling, which assigns a higher priority to a task with smaller laxity, has been known as an optimal preemptive scheduling algorithm on a single processor platform. However, its characteristics upon multiprocessor platforms have been little studied until now. Orthogonally, it has remained open how to efficiently schedule general task systems, including constrained deadline task systems, upon multiprocessors. Recent studies have introduced zero laxity (ZL) policy, which assigns a higher priority to a task with zero laxity, as a promising scheduling approach for such systems (e.g., EDZL). Towards understanding the importance of laxity in multiprocessor scheduling, this paper investigates the characteristics of ZL policy and presents the first ZL schedulability test for any work-conserving scheduling algorithm that employs this policy. It then investigates the characteristics of LLF scheduling, which also employs the ZL policy, and derives the first LLF-specific schedulability test on multiprocessors. It is shown that the proposed LLF test dominates the ZL test as well as the state-of-art EDZL test.**

I. INTRODUCTION

Real-time scheduling theory has been studied for satisfying timing constraints. In particular, scheduling policies for uniprocessor platforms have been extensively studied, and Earliest Deadline First (EDF) [1] and Deadline Monotonic (DM) [2] were developed as optimal dynamic- and static-priority scheduling policies. While uniprocessor scheduling has successfully matured over years, the same cannot be said about scheduling theory for multi-cores (multiprocessors).

Some multiprocessor studies in the past (e.g., [3], [4], [5]) have focused on adapting existing uniprocessor scheduling to multiprocessors, and some others have developed novel policies specific to multiprocessors (e.g., [6], [7], [8], [9], [10], [11], [12]). In spite of some significant achievements of these studies, many important scheduling problems continue to pose challenges, including the efficient scheduling of general task systems such as those in which task deadlines differ from their periods. We believe that one of the primary reasons for this lack of success is the sole focus on deadline satisfaction (or "urgency") by these existing approaches. When a task cannot be simultaneously scheduled on more than one processor at the same time ("parallelism" restriction), it becomes equally important to consider task "parallelism" when assigning priorities to tasks. Otherwise, a task may fail to meet its deadlines because the scheduler gave it processing capacity with more parallelism than it could utilize.

Considering that a job with smaller time to deadline is more urgent and a job with larger execution time has more parallelism restriction, one of the simple but effective ways to consider both urgency and parallelism is to assign the highest priority to any zero-laxity task, where laxity of a task at any time is defined as remaining time to deadline minus the amount of remaining execution. We denote this policy as the *ZL policy*, and any work-conserving[1] scheduling algorithm that employs this policy as a *ZL-based scheduling algorithm*. EDZL scheduling [13], [3], which globally (single run queue for all the processors) employs the EDF strategy until tasks have zero-laxity and the ZL policy, thereafter, is one example of a successful ZL-based scheduling algorithm; it dominates[2] global EDF [14] and it has been observed to be relatively more efficient in scheduling general task systems than many algorithms (see Figure 2). Likewise, Least Laxity First (LLF) scheduling [15], which globally assigns a higher priority to a task with lower laxity, is another example of a ZL-based scheduling algorithm that has shown promise in simulations (see Figure 2). Given the impact that the ZL policy tends to have on multiprocessor scheduling of general task systems, we provide the first general schedulability test applicable to any ZL-based scheduling algorithm in this paper. This test is a simple extension of the existing EDZL schedulability test [16].

LLF is an interesting ZL-based scheduling algorithm because, in addition to having good performance in simulations, it is known to be optimal for general task systems in the uniprocessor case [15], [17]. Further, unlike in EDZL, the ZL policy is implicit in LLF, suggesting a natural and close connection between ZL and LLF policies. These, combined with the fact that LLF multiprocessor scheduling has received little attention, have driven us to focus on it in this study. Although the aforementioned ZL schedulability test can also be applied to LLF, we derive a tighter LLF-

‡A corresponding author

[1]A work-conserving multiprocessor scheduling algorithm always schedules any unfinished, ready-to-execute task if there are available processors.

[2]Scheduling algorithm (test) A dominates B if any task set deemed schedulable by B is also deemed schedulable by A, but the vice-versa is not true.

specific test in this paper. For this purpose, we perform the following three steps.

1) We identify and define some properties associated with the LLF policy. We consider the scenario that a task misses its deadline at some time instant t_0, and compute the number of tasks that can have certain laxity values at certain time instants ahead of t_0. Lower bounds on these numbers that ensure the deadline miss are then used to define the properties. These properties are significant because they represent invariants (lower bounds) for highly dynamic system parameters (task laxity values).
2) We characterize these properties using conditions on worst-case higher-priority interference, because previous studies suggest that it is feasible to derive multiprocessor schedulability tests using such conditions (e.g., [16]).
3) Using upper bounds on the higher-priority interference, derived in this paper and shown to be tighter than previously known bounds, we finally derive the LLF-specific schedulability test.

To the best of our knowledge, this is the first LLF-specific schedulability test for multiprocessors, and we show that it dominates the state-of-art EDZL test [16] as well as the aforementioned ZL test[3].

In summary, this paper makes the following contributions. It proposes the first general schedulability test applicable to any ZL-based scheduling algorithm (Section III). It identifies and characterizes some laxity properties associated with the LLF policy (Section IV), and then derives the first LLF-specific schedulability test (Section V). This test is shown to dominate the state-of-art EDZL test as well as the ZL test derived here.

II. System Model

Task model. In this paper we assume a sporadic task model [18]. In this model, we specify a task τ_i as (T_i, C_i, D_i), where T_i is minimum separation, C_i is the worst-case execution time requirement, and D_i is the relative deadline. Task τ_i invokes a series of jobs, each separated from its predecessor by at least T_i time units. Further, we assume a constrained deadline task system, i.e., $C_i \leq D_i \leq T_i$ for each task τ_i. We also assume that a single job of a task cannot be executed in parallel.

In this paper we assume quantum-based time and without loss of generality let one time unit denote the quantum length. All task parameters are assumed to be specified as multiples of this quantum length.

We use $D_i(t)$ and $C_i(t)$ to denote the remaining time to deadline and the remaining execution time, respectively, of a job of τ_i at time t. Note that since we focus on constrained deadline task systems, these quantities are well-defined. We express that a job of τ_i is *active* at t when $C_i(t)$ is non-zero. We use $L_i(t)$ to denote the laxity of a job of τ_i at t, and then by definition we have $L_i(t) = D_i(t) - C_i(t)$. We denote the total number of tasks as n, and define system utilization by $U_{sys} = \sum_j \frac{C_j}{T_j}$ and system density by $D_{sys} = \sum_j \frac{C_j}{D_j}$.

Multiprocessor platform. We assume that the platform is comprised of m identical unit-capacity processors, and therefore restrict the system utilization U_{sys} to at most m. It has been previously shown that $U_{sys} \leq m$ is a necessary condition for feasibility of the task system considered here [6]. Like most existing studies in multiprocessor scheduling (for example, see [6]), we assume that the system does not incur any penalty when a job is preempted or when a job is migrated from one processor to another.

[3]This dominance relation only applies to the schedulability tests, and not to the algorithms themselves.

III. Schedulability Analysis for ZL-based Scheduling Algorithms

Recent studies have characterized the ZL policy and used it in EDZL schedulability analysis [16], but this analysis is EDZL-specific in that it cannot be directly used by any other ZL-based algorithm. In this section, building upon the EDZL analysis of [16], we derive schedulability conditions for any ZL-based scheduling algorithm.

A. Analysis on existing EDZL schedulability test

The EDZL schedulability test proposed in [16], uses the following two observations to capture characteristics of the ZL policy in EDZL scheduling. When a deadline miss occurs under EDZL, it must be true that

- *Observation A*: there exist at least $m + 1$ tasks which have zero or negative laxity.
- *Observation B*: there exists at least one task which has negative laxity.

In the above observations, when we use the term "task" it actually refers to some "active job" of that task. Nevertheless the usage is correct, because at any time instant there is at most one "active job" for any constrained deadline task. Note that Observation A is only a necessary condition for the deadline miss because it does not require the $m + 1$ tasks to have zero or negative laxity "at the same time". Unlike this however, Observation B is both necessary and sufficient. Nevertheless, Observation A is still relevant because the EDZL conditions derived in [16] to capture these observations are only necessary and not sufficient. As a result, schedulability tests that use both these observations are tighter than those that use only Observation B.

Observation A originates from the ZL policy. If a job misses its deadline at time t_1, then there must exist t_0 ($< t_1$) at which the job has zero or negative laxity but is not scheduled. This means, under EDZL scheduling, the instant t_0 will have at least m other jobs with zero or negative laxity. Observation B can be applied to any work-conserving scheduling algorithm in that it is impossible that a job misses its deadline without having negative laxity. Both these conditions are therefore necessarily true when a deadline miss occurs under EDZL. Furthermore, since both the conditions do not characterize any EDZL-specific properties except the ZL and work-conserving policies, they are also necessarily true when a deadline miss occurs under any ZL-based scheduling algorithm. Schedulability test for ZL-based scheduling algorithms can then be generally stated as follows:

Observation 1: A task system is schedulable by any ZL-based scheduling algorithm on m processors unless both Observation A and Observation B are satisfied.

B. ZL schedulability test

In order to check whether a task τ_k can have zero or negative laxity, existing approaches have used the concept of worst-case interference of higher-priority tasks on a job of task τ_k between its release and deadline. Following the notations similar to existing studies [19], [16], [20], we denote the total interference of a task τ_i on a task τ_k in interval $[t_a, t_b)$ as $\overline{I}_{k,i}(t_a, t_b)$. It represents the cumulative length of all intervals within $[t_a, t_b)$ in which τ_k is ready to execute and τ_i is executing while τ_k is not. The worst-case interference of τ_i on τ_k in any interval of length l is then defined as

$$I_{k,i}(l) = \max_t \overline{I}_{k,i}(t, t+l), \tag{1}$$

and the overall worst-case higher priority interference on τ_k is defined as

$$\sum_{i \neq k} I_{k,i}(l). \tag{2}$$

Note that the above equation over-estimates interference, because it does not consider the fact that the worst-case interference scenario for each task may occur in different time intervals. It is known that computing $I_{k,i}(l)$ precisely is hard, and therefore existing approaches have used an upper bound that is valid under any work-conserving scheduling algorithm [21], [20]. These studies describe the job-release pattern corresponding to the largest workload of a task τ_i that can interfere with a task τ_k. This pattern is depicted in Figure 1. Given an interval $[t_a, t_b)$ of length l, the first job of τ_i starts at t_a and ends at $t_a + C_i$. Here $t_a + C_i$ is also the deadline of the first job. Thereafter, jobs are released and scheduled as soon as possible. We denote by $\eta_i(l)$ the number of jobs of τ_i that can execute completely within the interval of interest (including the first job).

$$\eta_i(l) = \left\lfloor \frac{l - (C_i + T_i - D_i)}{T_i} \right\rfloor + 1 = \left\lfloor \frac{l + D_i - C_i}{T_i} \right\rfloor \tag{3}$$

The contribution of the last job can then be bounded by $\min(C_i, l + D_i - C_i - \eta_i(l) \cdot T_i)$. The maximum interference of a task τ_i on a task τ_k during an interval of length l under any work-conserving scheduling algorithm (denoted by $I^{\mathsf{WC}}_{k,i}(l)$) is therefore

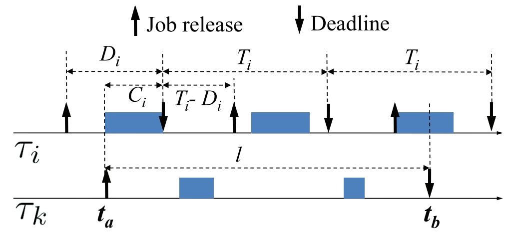


Figure 1. Situation when the maximum interference occurs under any work-conserving algorithm.

$$I^{\mathsf{WC}}_{k,i}(l) = \eta_i(l) \cdot C_i + \min(C_i, l + D_i - C_i - \eta_i(l) \cdot T_i) \tag{4}$$

Using $I^{\mathsf{WC}}_{k,i}(l)$, the following lemma introduces a condition for the case when jobs have zero or negative laxity under any ZL-based scheduling algorithm (extension of Theorem 7 in [16]):

Lemma 1: If task τ_k has zero or negative laxity, then

$$\sum_{i \neq k} \min(I^{\mathsf{WC}}_{k,i}(D_k), D_k - C_k) \geq m \cdot (D_k - C_k) \tag{5}$$

Proof: Same as proof of Theorem 7 in [16]. ■

In the above lemma, Lemma 4 in [19] (i.e., $\sum_{i \neq k} I^{\mathsf{WC}}_{k,i}(t) \geq m \cdot x \iff \sum_{i \neq k} \min\{I^{\mathsf{WC}}_{k,i}(t), x\} \geq m \cdot x$) is applied in order to tighten the condition.

Similarly, the following lemma holds for tasks with negative laxity under ZL-based scheduling algorithms (extension of Theorem 7 in [16]):

Lemma 2: If task τ_k has negative laxity, then[4]

1) $\sum_{i \neq k} \min(I^{\mathsf{WC}}_{k,i}(D_k), D_k - C_k) > m \cdot (D_k - C_k)$, or (6)

2) $\sum_{i \neq k} \min(I^{\mathsf{WC}}_{k,i}(D_k), D_k - C_k) = m \cdot (D_k - C_k)$

and $\forall i \neq k : D_k - C_k < I^{\mathsf{WC}}_{k,i}(D_k)$ (7)

Proof: Same as proof of Theorem 7 in [16]. ■

Using Lemmas 1 and 2, we formally express Observation 1 as follows:

Theorem 1 (ZL Schedulability): A task set is schedulable by any ZL-based scheduling algorithm unless both 1) and 2) are true:

1) There are at least $m + 1$ tasks τ_k satisfying Eq. (5).

2) There is at least one task τ_k satisfying either Eq. (6) or Eq. (7).

As LLF scheduling employs the ZL policy implicitly, Theorem 1 can be used as a schedulability test for LLF as well. However, since LLF accommodates some additional

[4]Theorem 7 in [16] only includes Eq. (6), but it is corrected in [22].

properties in addition to ZL policy, it would be more interesting to derive another LLF-specific schedulability condition by generalizing the above observations used in the ZL test. We therefore characterize LLF-specific properties in Section IV, and in Section V introduce a new LLF schedulability test based on these properties.

IV. Characteristics of LLF

In this section we first motivate our choice of LLF policy among all ZL-based scheduling algorithms through a discussion on its performance. We then characterize the LLF-specific properties associated with a deadline miss, which will serve as a basis for deriving a LLF-specific schedulability test in the next section.

A. Scheduling performance of LLF

Substantial studies have been undertaken on multiprocessor scheduling theory (e.g., [6], [7], [8], [10], [11], [12]), including those that have introduced optimal scheduling algorithms (e.g., Pfair [6]) for certain classes of task systems. However, it has been observed that their performance degrades significantly when considering more general task systems, such as constrained deadline task systems considered here. Although optimal scheduling of such general task systems has been shown to be impossible [23], we cannot rule out the existence of algorithms which are more efficient than the aforementioned ones. We believe the challenges involved in efficiently scheduling such task systems have not yet been well-understood ("urgency" vs. "parallelism" issue), and this is the primary reason for the lack of success.

ZL policy, as discussed in the introduction, seems to be a simple and effective mechanism in handling the twin-issues of deadline satisfaction and parallelism restriction. For example, EDZL studies [13], [3], [16] have demonstrated the impact of the ZL policy on schedulability of constrained deadline task systems. LLF strategy, which is yet another ZL-based scheduling algorithm like EDZL, has so far received very little attention in the literature on multiprocessor scheduling [24]. Study [24] shows that a task set feasible on m speed-1 processors is schedulable under LLF scheduling on both (a) m speed-(2-1/m) processors and (b) m+$O(\log(\max_i C_i/\min_i C_i))$ speed-1 processors. To use this test as a schedulability test for LLF however, a sufficient feasibility test is required. To our best knowledge, the only known technique to check feasibility is to use schedulability tests such as those described above. This means that any LLF test derived from [24] will be only as good as these previously known schedulability tests.

One may then wonder how good the LLF strategy is for scheduling constrained deadline task systems on multiprocessors. Figure 2 shows simulation results over a variety of scheduling algorithms (see figure caption for a detailed description of the simulation setting). Figure 2(a) shows that when the system density is no greater than m (i.e., $D_{sys} \leq m$), Pfair[5], LLF, and EDZL, all perform well without regard to the number of processors, and EDF however performs worse as m increases. It is worth noting that in spite of the non-optimality of LLF and EDZL as opposed to the optimality of Pfair for task systems with $D_{sys} \leq m$, the performance of LLF and EDZL is very close to that of Pfair in our simulations. For example, LLF fails to schedule 0.05% $(m = 2)$, 0.007% $(m = 4)$, and 0% $(m > 4)$, and EDZL fails to schedule 0.16% $(m = 2)$, 0.06% $(m = 4)$, and 0% $(m > 4)$. Figure 2(b) shows that when the system density is greater than m (i.e., $D_{sys} > m$), Pfair, EDZL, and LLF show different behaviors. LLF significantly outperforms the others on average. In particular, Pfair performs quite poorly in this case. In general, ZL-based algorithms (LLF, EDZL, ZL) perform much better than non-ZL-based ones (Pfair, EDF).

These simulation results indicate that LLF is quite effective in scheduling constrained deadline task systems on multiprocessors, in particular, relatively more effective when $D_{sys} > m$. Hence, in this paper, we aim to understand LLF multiprocessor scheduling and introduce the first LLF-specific schedulability test for multiprocessor platforms. Note that we do not assume a specific tie-breaking rule with the LLF algorithm, so that our proposed schedulability test is generally applicable.

B. Observation from deadline miss under LLF

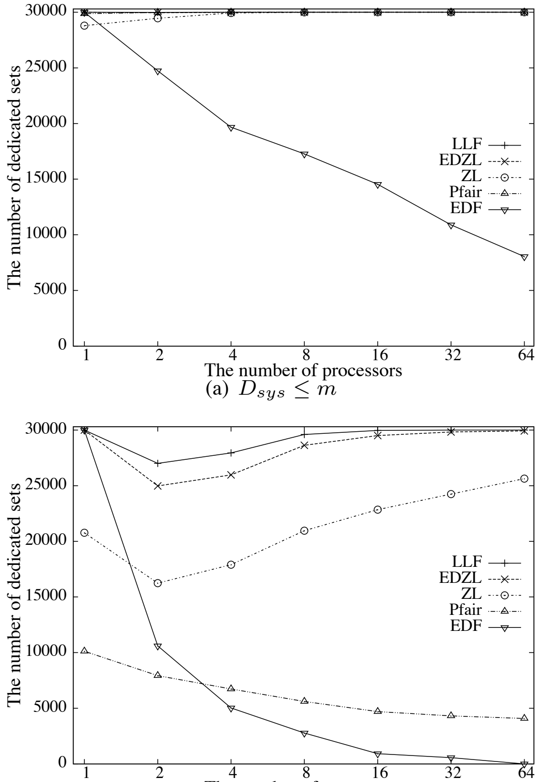
In this subsection we characterize LLF-specific properties related to a missed task deadline. We first investigate necessary conditions at each instant before the deadline miss, and show that the conditions depend on various parameters of tasks like laxity values, release times, and finishing times. Then, we abstract the conditions such that they only depend on the laxity values of tasks. These conditions form the basis for the new LLF schedulability test proposed in Section V.

Let $S_\theta(t)$ denote a set of tasks whose jobs have a laxity of θ at time instant t, and let $N_\theta(t)$ denote the size of $S_\theta(t)$. Note that $S_{-1}(t)$ is defined to represent a set of tasks whose jobs have negative laxity. Suppose there is a task that misses a deadline, and let t_0 denote the first time instant before the deadline miss when there is a task with negative laxity. That is, t_0 is the first time instant such that $S_{-1}(t_0) \neq \emptyset$. Note that $S_\theta(t)$, $N_\theta(t)$, and t_0, will be used in the rest of the paper, including in lemmas, definitions, and theorems, without restating what they stand for.

Let us consider what would happen at $t_0 - 1$. In fact, there must be more than m tasks with zero laxity, and we present this observation formally as follows:

Observation 2: The following holds under LLF scheduling:

[5]Pfair is originally defined for implicit-deadline task systems such that each task's period (equal to deadline) is split into sub-deadlines with execution time of one unit. To adapt Pfair for constrained deadline task systems, we split each task's deadline into sub-deadlines.



(a) $D_{sys} \leq m$

(b) $D_{sys} > m$

Figure 2. These figures show simulation results of various scheduling algorithms over constrained deadline tasks on a different number of processors. Such algorithms include LLF, EDZL, ZL, Pfair, and EDF, where ZL algorithm gives the highest priority to tasks with zero laxity (ZL policy), and gives randomly chosen static priority to tasks with positive laxity. Here y-axis means the number of task sets that are deemed schedulable by simulation. We perform simulation over $30,000$ task sets for both $D_{sys} \leq m$ and $D_{sys} > m$ during the first $100,000$ time units for tractability. The simulation environment is described in more details in our technical report [25].

$$[A_1]: \qquad N_0(t_0-1) > m.$$

Note that the above observation holds generally for all ZL-based scheduling algorithms. The next step is to consider what would happen at t_0-2 depending on what happens at t_0-1. We first consider a case where there is no job released or finished at t_0-1. By definition, there are $N_0(t_0-2)$ tasks with zero laxity at t_0-2. We observe that $N_0(t_0-2) \leq m$ because otherwise t_0-1 is the first instant when there is a task with negative laxity. Thus, considering zero-laxity tasks have the highest priority under LLF scheduling, $N_0(t_0-2)$ zero-laxity tasks will be all scheduled in $[t_0-2, t_0-1)$, and they will continue to have zero laxity at t_0-1. In addition, some of the one-laxity tasks can be scheduled, and all the remaining tasks will not be scheduled. That is, $m-N_0(t_0-2)$ one-laxity tasks will be scheduled at t_0-2 together with $N_0(t_0-2)$ zero-laxity tasks. Hence, among $N_1(t_0-2)$ one-laxity tasks at t_0-2, $N_1(t_0-2)-(m-N_0(t_0-2))$ tasks will not be scheduled at t_0-2, and their laxity will become zero at t_0-1. Here we observe that $N_1(t_0-2)-(m-N_0(t_0-2)) > 0$. If it is not true, all tasks with one or zero laxity at t_0-2 are scheduled at $[t_0-2, t_0-1)$, and then $[A_1]$ in Observation 2 does not hold. So the number of zero-laxity tasks at t_0-1 is given by

$$\begin{aligned} N_0(t_0-1) &= N_0(t_0-2)+N_1(t_0-2)-(m-N_0(t_0-2)) \\ &= 2\cdot N_0(t_0-2)+N_1(t_0-2)-m \\ &> m. \end{aligned} \tag{8}$$

Extending Observation 2, we present this observation formally as follows:

Observation 3: If there is no job released or finished at t_0-1, the following holds under LLF scheduling:

$$[A_2]: \qquad 2\cdot N_0(t_0-2)+N_1(t_0-2) > 2\cdot m.$$

It is worth noting that the above observation is specific to LLF, and in particular, does not necessarily hold for other ZL-based scheduling algorithms. Now we consider the general case where there are jobs released and/or finished at t_0-1. We denote $Z_\theta(t_0-x)$ as the number of tasks whose jobs are released at t_0-x+1 with a laxity of θ. We also denote $W_\theta(t_0-x)$ as the number of tasks whose jobs are finished at t_0-x+1 and have a laxity of θ at t_0-x. If $N_0(t_0-2)+N_1(t_0-2)$ is larger than m, we can calculate $N_0(t_0-1)$ similarly as in Eq. (8):

$$\begin{aligned} N_0(t_0-1) = 2\cdot N_0(t_0-2)+N_1(t_0-2)-m \\ +Z_0(t_0-2)-W_0(t_0-2) \qquad > m. \end{aligned} \tag{9}$$

If $N_0(t_0-2)+N_1(t_0-2)$ is not larger than m, all tasks in $S_0(t_0-2)$ and $S_1(t_0-2)$ are scheduled and keep their laxity values at t_0-1. This means no task in $S_1(t_0-2)$ will belong to $S_0(t_0-1)$, and the following is true.

$$\begin{aligned} &N_0(t_0-1) = N_0(t_0-2)+Z_0(t_0-2)-W_0(t_0-2) > m \\ &\Longrightarrow 2\cdot N_0(t_0-2)+2\cdot\{Z_0(t_0-2)-W_0(t_0-2)\} > 2\cdot m \end{aligned} \tag{10}$$

To summarize, we present Eqs. (9) and (10) using sufficient conditions as follows:

Observation 4: The following holds under LLF scheduling:

$$\begin{aligned} [A_2']: \qquad & 2\cdot N_0(t_0-2)+N_1(t_0-2)+ \\ & (1+\mathbf{1}_{t_0-2})\{Z_0(t_0-2)-W_0(t_0-2)\} \\ & > 2\cdot m \end{aligned}$$

where

$$\mathbf{1}_{t_0-x} = \begin{cases} 0 & \text{, if } \sum_{j=0}^{x-1} N_j(t_0-x) > m \\ 1 & \text{, if } \sum_{j=0}^{x-1} N_j(t_0-x) \leq m \end{cases}$$

Now we wish to generalize the above observation for all time instants before t_0, and present the following lemma.

Lemma 3: Each of the following holds under LLF scheduling for $x = 1, 2, 3, ..., \infty$:

$$[A'_x]: \quad \sum_{j=0}^{x-1}(x-j)\cdot N_j(t_0-x) + \sum_{k=2}^{x}\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\} \cdot\{Z_j(t_0-k)-W_j(t_0-k)\} > x\cdot m$$

Proof: The basic idea of the proof is to use mathematical induction, and we consider the following two cases for each inductive step at t: $\mathbf{1}_t = 1$ and $\mathbf{1}_t = 0$. Detailed proof is given in the Appendix. ■

Eq. $[A'_x]$ in Lemma 3 can be further simplified by eliminating the contribution of terms $Z_j(t_0-k)$ and $W_j(t_0-k)$. For this purpose, consider the following definition of *perceived laxity* of a task.

Definition 1: The *perceived laxity* of a task τ_k at t (denoted by $\bar{L}_k(t)$) is defined as follows:

$$\bar{L}_k(t) = \begin{cases} L_k(t)\ (= D_k(t)-C_k(t)) & \text{, if } t \geq t_0 - D_k \\ D_k - C_k & \text{, if } t < t_0 - D_k \end{cases} \quad (11)$$

Just as we defined $\bar{L}_k(t)$ corresponding to $L_k(t)$, we now define $\bar{N}_j(t)$ corresponding to $N_j(t)$:

Definition 2: Let $\bar{N}_j(t)$ denote the number of tasks with a perceived laxity of j at time instant t under LLF scheduling.

Note that LLF still uses the actual laxity of tasks to assign priorities; the perceived laxity is only used in the schedulability test. Using Definition 1 and 2, we do not have to care about the contribution of terms $Z_j(t_0-k)$ and $W_j(t_0-k)$. In particular, the following theorem shows that Eq. $[A'_x]$ can be safely simplified if we employ $\bar{N}_j(t)$ instead of $N_j(t)$.

Theorem 2: Each of the following holds under LLF scheduling for $x = 1, 2, 3, ..., \infty$:

$$[A_x]: \quad \sum_{j=0}^{x-1}(x-j)\cdot\bar{N}_j(t_0-x) > x\cdot m$$

Proof: The basic idea of the proof is to show that the LHS of $[A_x]$ is equal to or larger than the LHS of $[A'_x]$ in Lemma 3. Then $[A_x]$ is a necessary condition of $[A'_x]$ and therefore this theorem holds.

To prove that the LHS of $[A_x]$ is equal to or larger than the LHS of $[A'_x]$, we investigate how much each task contributes to these quantities. Detailed proof is given in the Appendix. ■

V. SCHEDULABILITY TEST FOR LLF

In the previous section, we derived necessary conditions for a task to have negative laxity at t_0 under LLF scheduling, in terms of conditions on the number of tasks with certain laxity values prior to t_0 ($\bar{N}_j(t_0-x)$). In this section, we investigate how to incorporate those conditions into a schedulability test for LLF. For this purpose, we first introduce a new worst-case task interference bound for LLF scheduling. Based on this interference bound, we derive upper-bounds for the terms $\bar{N}_j(t_0-x)$. Then, we propose a new schedulability test for LLF, and analyze its time complexity.

A. Worst-case Interference function for LLF

To upper-bound $\bar{N}_j(t_0-x)$-terms in Theorem 2, we will derive necessary conditions for a task to have a certain laxity value at a certain time instant ahead of its deadline. To do this, in this subsection, we derive the worst-case interference bound of a task τ_i on a task τ_k in a time interval $[t_a, t_b)$, where t_a is the release time of τ_k's job and t_b is some time instant no later than the deadline of the job (i.e., $t_b \leq t_a + D_k$). Although the previously known interference bound for any work-conserving algorithm (Eq. (4)) can also be used for LLF, we present a tighter LLF-specific interference bound in this section.

Consider the interference pattern shown in Figure 1 corresponding to the term $I_{k,i}^{\mathsf{WC}}(D_k)$; here t_b is the deadline of τ_k's job (i.e., $t_b = t_a + D_k$). Suppose the carry-out job of task τ_i in the figure[6] has a laxity of x or greater until t_b. Then this job cannot interfere with the execution of task τ_k in the interval $[t_b-x, t_b)$ under LLF scheduling. This follows from the fact that by definition τ_k has a laxity of at most $x-1$ at $t_b - t \in [t_b - x, t_b)$ if it has remaining executions at $t_b - t$. That is, $D_k(t_b-t) = t$, $C_k(t_b-t) \geq 1$, and $L_k(t_b-t) \leq t-1 \leq x-1$. Thus the worst-case interference pattern of task τ_i on task τ_k in $[t_a, t_a + D_k)$ is as shown in Figure 3. For the interference function to be useful in bounding $\bar{N}_j(t_0-x)$ for arbitrary values of j and t_0-x, it is necessary to define the function even for cases when the considered interval has length smaller than D_k and τ_k has some arbitrary laxity value θ or smaller at the end of the interval. Suppose t_b is prior to the deadline of τ_k's job (i.e., $t_b \leq t_a + D_k$) and τ_i has a laxity of $\theta+1+x$ until t_b. Then again, τ_i cannot interfere with τ_k in the interval $[t_b-x, t_b)$, because τ_k has a laxity of at most $\theta+x$ in that interval. Thus the worst-case interference pattern of task τ_i on task τ_k in $[t_a, t_b)$ is as shown in Figure 4, and we formally express the pattern as a function of l and θ, where l is the interval length and θ is a laxity value which task τ_k has at the end of the interval.

[6]Here a carry-out job means it is released within the given interval, but its deadline is after the interval.

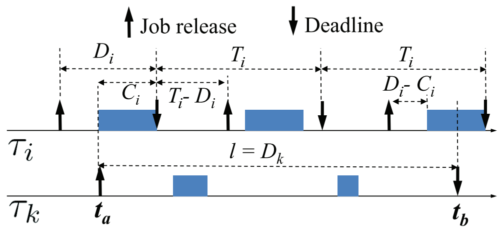


Figure 3. Situation when the maximum interference occurs under LLF scheduling with interval length D_k.

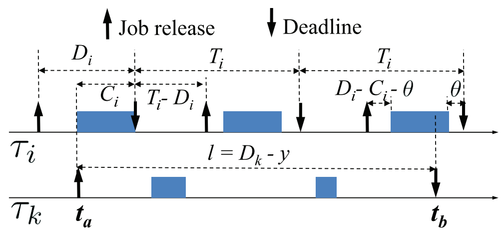


Figure 4. Situation when the maximum interference occurs under LLF scheduling with at most θ laxity y time units ahead of D_k.

$$\begin{aligned} I^{\mathsf{LLF}}_{k,i}(l,\theta) &= \eta_i(l)\cdot C_i + \max\{0, \min(C_i, l + D_i - C_i - \eta_i(l)\cdot T_i \\ &\qquad -(D_i - C_i - \max\{0, \min(D_i - C_i, \theta)\})\} \\ &= \eta_i(l)\cdot C_i + \max\{0, \min(C_i, l - \eta_i(l)\cdot T_i \\ &\qquad + \max\{0, \min(D_i - C_i, \theta)\})\}. \end{aligned} \tag{12}$$

Here $\eta_i(l)$ is defined as in Eq. (3). The above equation is similar to Eq. (4), but the difference is that the interference of the carry-out job is deducted by at most $D_i - C_i - \theta$ as shown in Figure 4.

B. Laxity and interference relation

Lemmas 1 and 2 use the interference function $I^{\mathsf{WC}}_{k,i}$ to determine whether a task τ_k can have zero or negative laxity under any ZL-based scheduling algorithm. To be able to bound $\bar{N}_j(t_0 - x)$ for all possible values of j and $t_0 - x$ in Theorem 2, we now generalize these lemmas for different possible laxity values of task τ_k and for different possible interval lengths under LLF scheduling.

Lemma 4: If a task τ_k has a laxity of θ or less at y time units ahead of its deadline, then the following inequality holds:

$$\sum_{i\neq k} I^{\mathsf{LLF}}_{k,i}(D_k - y, \theta) \geq m\cdot(D_k - C_k - \theta) \tag{13}$$

By Lemma 4 in [19], the above inequality can be further tightened as:

$$\sum_{i\neq k} \min(I^{\mathsf{LLF}}_{k,i}(D_k - y,\theta), D_k - C_k - \theta) \geq m\cdot(D_k - C_k - \theta) \tag{14}$$

Proof: We prove this lemma by contraposition. That is, assuming $\sum_{i\neq k} I^{\mathsf{LLF}}_{k,i}(D_k - y,\theta) < m\cdot(D_k - C_k - \theta)$, we prove that task τ_k must have a laxity larger than θ at y time units ahead of its deadline.

The total interference to task τ_k is less than $m\cdot(D_k - C_k - \theta)$, and τ_k cannot execute at some time instant only when m other tasks execute at that instant. Thus, during the interval $[t_a, t_a + D_k - y)$ (where t_a is the release time of τ_k's job), τ_k is prevented from executing for less than $m\cdot(D_k - C_k - \theta)/m$ time units due to interference. This means that τ_k executes for at least $D_k - y - (D_k - C_k - \theta - 1) = C_k + \theta + 1 - y$ time units in that interval. Then, the laxity of τ_k at $t_a + D_k - y$ can be computed as follows:

$$\begin{aligned} D_k(t_a + D_k - y) &= & y \\ C_k(t_a + D_k - y) &\leq & C_k - (C_k + \theta + 1 - y) = y - \theta - 1 \\ L_k(t_a + D_k - y) &\geq & y - (y - \theta - 1) = \theta + 1 \end{aligned} \tag{15}$$

Thus, task τ_k's laxity at y time units ahead of its deadline is larger than θ, and this conclude the proof. ■

For a given time to deadline y ($\leq D_k$), let us define the following indicator function $\delta^*_k(\theta, y)$ for a task τ_k based on Lemma 4. This function indicates whether τ_k can reach a laxity of exactly θ at y time units ahead of its deadline.

$$\delta^*_k(\theta,y) = \begin{cases} 1, & \text{if this is the smallest } \theta \\ & \text{for which Eq. (14) is true.} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Also, to be consistent with our definition of $\bar{N}_j(t_0 - x)$, let us define $\delta^*_k(\theta, y)$ for $y > D_k$ as follows.

$$\delta^*_k(\theta,y) = \begin{cases} 1, & \text{if } \theta = D_k - C_k. \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Incorporating $\delta^*_k(\theta, y)$ into Theorem 2 we get the following lemma.

Lemma 5: Each of the following holds under LLF scheduling for $x = 1, 2, 3, ..., \infty$.

$$[B_x]: \qquad \sum_{j=0}^{x-1}(x - j)\sum_k \delta^*_k(j, x) > x\cdot m$$

Proof: We show that the LHS of $[B_x]$ is equal to or larger than the LHS of $[A_x]$ in Theorem 2 for $x = 1, 2, 3, ..., \infty$. Then $[B_x]$ is a necessary condition of $[A_x]$ and the lemma directly follows from Theorem 2.

We investigate how much individual tasks contribute to the LHS of $[A_x]$ in Theorem 2 and to that of $[B_x]$ in Lemma 5. Then, we prove that the contribution of a task τ_k to the LHS of $[B_x]$ is always equal to or larger than that to the LHS of $[A_x]$. We denote the LHS of $[A_x]$ as (A), and the LHS of $[B_x]$ as (B). We consider two cases depending on the value of x.

(Case 1) $t_0 - x$, where $x = 1, 2, ..., D_k$.

Suppose task τ_k has a laxity of θ' at $t_0 - x$. It then contributes $x - \theta'$ to (A). Recall that $\delta^*_k(\theta, x) = 1$ means τ_k can reach a laxity of θ at x time units ahead of its deadline, but it cannot reach a laxity of less than θ. This means that condition $\delta^*_k(\theta, x) = 1$ is necessary for τ_k to have a laxity of θ or less at x time units ahead of its deadline. Now, since τ_k has a laxity of θ' at $t_0 - x$, it holds that $\delta^*_k(\theta, x) = 1$ for some θ such that $\theta \le \theta'$. But then the contribution of τ_k to (B) is exactly $x - \theta$, which is at least as much as its contribution to (A). Thus, we can conclude that the contribution of any task to (B) is equal to or larger than that to (A) in this case.

(Case 2) $t_0 - x$, where $x = D_k + 1, D_k + 2, ..., \infty$.

By definition of $\delta^*_k(\theta, x)$ for $x > D_k$, task τ_k contributes $x - D_k + C_k$ to (B). Similarly by definition of perceived laxity for time instants before $t_0 - D_k$, task τ_k contributes $x - D_k + C_k$ to (A). Thus, the contribution to (A) and (B) are identical.

Finally, using the results of (Case 1) and (Case 2), we can conclude that the contribution of any task to (B) is larger than or equal to that of its contribution to (A). This concludes the proof. ∎

Here note that $[B_x]$ in Lemma 5, unlike $[A_x]$ in Theorem 2, only depends on the task parameters and nothing else; in particular it is independent of time instant t_0.

C. LLF schedulability test

Lemma 5 presents the necessary conditions for a deadline miss under LLF scheduling. Recall that these conditions are derived using constraints on the number of tasks with a certain laxity value at time instants $t_0 - 1, t_0 - 2, ...$, where t_0 denotes the time instant when there exists at least one task which has negative laxity. At t_0 we know that there exists at least one task with a negative laxity (Observation B). Therefore the conditions in Lemma 5 can be further augmented with one more necessary condition characterizing the negative laxity task at t_0. Similar to the ZL schedulability test in Theorem 1, we use Lemma 2 for this condition, replacing $I^{\mathsf{WC}}_{k,i}(D_k)$ with $I^{\mathsf{LLF}}_{k,i}(D_k, -1)$ in Eq. (12). Thus, combining all these observations, we formally express our LLF schedulability test as follows:

Theorem 3 (LLF schedulability): A task set is schedulable by LLF unless all the below statements ($[B_0]$, $[B_1]$, ..., $[B_{D_{max}}]$) are true, where $D_{max} = \max\{D_k\}$.

$[B_0]$ There is at least one task τ_k satisfying either Eq. (6) or Eq. (7), where $I^{\mathsf{WC}}_{k,i}(D_k)$ is replaced with $I^{\mathsf{LLF}}_{k,i}(D_k, -1)$.

$$[B_x]: \qquad \sum_{j=0}^{x-1} (x - j) \sum_k \delta^*_k(j, x) > x \cdot m, \quad \text{where } x = 1, 2, ..., D_{max}$$

Proof: This theorem holds from Lemmas 2 and 5. The difference between Lemma 5 and this theorem is in the range of x for $[B_x]$. Since satisfying $[B_x]$ in a limited range of x is a necessary condition for satisfying it in a more general range of x, correctness of this theorem holds trivially. Nevertheless, we now show that there is no need to investigate conditions $[B_x]$ for $x > D_{max}$. That is, assuming $[B_x]$ holds for all $x \le D_{max}$, we show that $[B_x]$ holds for all $x > D_{max}$ by mathematical induction.

(The basis) $[B_x]$ holds for all $x \le D_{max}$.

(The inductive step) We will prove that if $[B_x]$ holds, then $[B_{x+1}]$ also holds for $x \ge D_{max}$. Since $x \ge D_k$ for all τ_k, only $\delta^*_k(\theta = D_k - C_k, y)$ terms are equal to 1 for both $y = x$ and $y = x + 1$. Thus, the LHS of $[B_{x+1}]$ is increased by n (the number of tasks) compared to that of $[B_x]$, while the RHS of $[B_{x+1}]$ is increased by m (the number of processors). It holds that $n > m$ to meet $[B_1]$ is true, so $[B_{x+1}]$ is also true.

We conclude that we do not need to investigate conditions $[B_x]$ for $x > D_{max}$. ∎

It is not known whether there exists any dominance relationship between LLF and EDZL scheduling algorithms, but our LLF schedulability test described in Theorem 3 dominates the state-of-art EDZL schedulability test in [16], as well as the ZL test proposed in Theorem 1. To prove this dominance, we first prove that interference function $I^{\mathsf{LLF}}_{k,i}(l, \theta)$ for $\theta \le 0$ is equal to or less than the corresponding interference function under EDZL as described in [16], [20].

Lemma 6: $I^{\mathsf{LLF}}_{k,i}(l, \theta) \le I^{\mathsf{EDZL}}_{k,i}(l)$ for all values of l and $\theta \le 0$, where $I^{\mathsf{EDZL}}_{k,i}$ is defined as follows (from [16], [20]):

$$I^{\mathsf{EDZL}}_{k,i}(l) = \left\lfloor \frac{l}{T_i} \right\rfloor C_i + \min(C_i, l - \left\lfloor \frac{l}{T_i} \right\rfloor T_i), \qquad (18)$$

Proof: Due to the space limit, we refer readers to our technical report [25]. ∎

The following theorem then states the dominance relationship between schedulability tests.

Theorem 4: The LLF schedulability test in Theorem 3 dominates the EDZL test in [16] and the ZL test in Theorem 1.

Proof: The EDZL (and ZL) schedulability test is equivalent to the first two necessary conditions $[B_0]$ and $[B_1]$ where only $I^{\mathsf{LLF}}_{k,i}(l, \theta)$ for $\theta \le 0$ is used. Thus, this theorem holds from Lemma 6. ∎

In the above theorem we did not consider the iterative test method for EDZL [16]. This method computes each task's slack (minimum time interval between the latest finishing time and the deadline), and uses it to reduce the interference from carry-in jobs[7] of tasks. Since this technique can also be applied orthogonally to our LLF schedulability test, the aforementioned dominance relationship nevertheless holds.

[7] A carry-in job is released before the start of the interval, but may execute within the interval.

Remaining time to deadline (y)	Laxity (θ)				
	0	1	...	$D_k - C_k - 1$	$D_k - C_k$
0					
1	$\surd$				
2	$\surd$	$\surd$			
...	$\surd$	$\surd$	...		
$D_k - C_k$	$\surd$	$\surd$	$\surd$	$\surd$	
$D_k - C_k + 1$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
...	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C_k	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$C_k + 1$		$\surd$	$\surd$	$\surd$	$\surd$
...			...	$\surd$	$\surd$
$D_k - 1$				$\surd$	$\surd$
D_k					$\surd$

Table I
A SET OF POSSIBLE NON-NEGATIVE LAXITY VALUES ACCORDING TO REMAINING TIME TO DEADLINE

D. *Complexity of LLF schedulability analysis*

When we apply the ZL (as well as EDZL) schedulability test, the calculation of the LHS of Eq. (5) for a given task τ_k has complexity $O(n)$. We need to calculate the LHS of Eq. (5) for all tasks in the worst case, and then the ZL (as well as EDZL) schedulability test has the complexity $O(n^2)$.

Similarly, when we test schedulability of LLF, calculation of the LHS of Eq. (14) for a given τ_k, θ, and y, has complexity $O(n)$. Given a task τ_k, we need to calculate Eq. (14) for all pairs (θ, y) marked as $\surd$ in Table I in the worst-case, where the number of pairs is $O(D_k \cdot (D_k - C_k))$. Therefore, overall, the LLF schedulability test has complexity $O(n^2 \cdot \max_k D_k \cdot (D_k - C_k))$. In fact, this complexity can be reduced to $O(n^2 \cdot \max_k(D_k))$, if we take advantage of properties associated with $\delta_k^*(\theta, y)$. Details how to achieve the complexity are presented in our technical report [25].

Note that the structure of LLF schedulability test consists of a set of necessary conditions for a deadline miss. The correctness of our LLF test holds even if we investigate any partial subset of the $[B_i]$ conditions in Theorem 3. For example, if we only consider $[B_0]$ and $[B_1]$, the schedulability test has similar complexity to that of the ZL test. Thus there exists a trade-off between complexity and schedulability in terms of how many necessary conditions $[B_i]$ are checked.

VI. CONCLUSION

In this work we have identified some properties of LLF scheduling, over and above those associated with the zero-laxity (ZL) policy. A successful characterization of these properties using worst-case higher priority interference has led to the first LLF-specific schedulability test for unit-capacity multiprocessor platforms. Dominance of this test over previously known tests for ZL-based algorithms has also been established.

While LLF is effective in terms of schedulability, a large number of preemptions is the main barrier to its practical use. LLF itself cannot avoid frequent preemptions; for example, when two jobs have the same laxity, they repeatedly preempt each other under LLF. We can however reduce the number of preemptions if we incorporate some tie-breaking rules into LLF. For example, by executing jobs that have the same laxity in EDF order, we can prevent them from repeatedly preempting each other. In the future, we will consider such tie-breaking rules for the LLF scheduling algorithm, and develop corresponding schedulability tests, based on the test developed in this paper. In spite of such modifications, relative performance of the LLF test may degrade if preemption costs are considered. Therefore, we also plan to develop preemption-aware LLF analysis, so that our results can be compared with other algorithms (e.g., EDZL) under more practical environments.

Another direction of future work is to evaluate and understand how and why performance of LLF and other online algorithms degrades for various task systems such as implicit, constrained, and arbitrary deadline task systems (e.g., [23] for constrained deadline task systems). Based on understanding how urgency and parallelism constraints influence multiprocessor schedulability, we plan to design more efficient scheduling algorithms in multiprocessor platforms.

ACKNOWLEDGEMENT

The authors are grateful to the anonymous reviewers and the shepherd for helpful comments.

This work was supported in part by the IT R&D Program of MKE/KEIT [2010-KI002090, Development of Technology Base for Trustworthy Computing], Basic Research Laboratory (BRL) Program (2009-0086964) and Basic Science Research Program (2010-0006650) through the National Research Foundation of Korea (NRF) funded by the Korea Government (MEST), KAIST-Microsoft Research Collaboration Center, KAIST ICC, and KI-DCS grants.

This work was also partially funded by the Portuguese Science and Technology Foundation (FCT), the European Commission (ARTISTDesign), the ARTEMIS-JU (RECOMP), and the Luso-American Development Foundation (FLAD).

APPENDIX

A. *Proof of Lemma 3*

Proof: The basic idea is to prove the lemma by mathematical induction.

(Basis) Conditions of $t_0 - 1$ and $t_0 - 2$ ($[A'_1]$ and $[A'_2]$) are true by Observations 2 and 4.

(Inductive step) We wish to prove the following: for all $x \geq 2$, if the condition of $t_0 - x$ ($[A'_x]$) is true, then the condition of $t_0 - (x+1)$ ($[A'_{x+1}]$) is also true. We consider two cases depending on the value of $\mathbf{1}_{t_0-x-1}$.

(Case 1) Assume

$$\mathbf{1}_{t_0-x-1} = 1 \iff \sum_{j=0}^{x} N_j(t_0 - x - 1) \leq m. \tag{19}$$

All tasks in $S_j(t_0 - x - 1)$ for $0 \leq j < x$ are serviced in $[t_0 - x - 1, t_0 - x)$, and thus $L_k(t_0 - x - 1) = L_k(t_0 - x)$

for all tasks $\tau_k \in S_j(t_0-x-1)$ where $0 \le j < x$. So, the following relationship holds:

$$N_j(t_0-x) = N_j(t_0-x-1) + Z_j(t_0-x-1) - W_j(t_0-x-1) \quad ,\forall 0 \le j < x \tag{20}$$

Using the above equation in $[A'_x]$ we get:

$$\sum_{j=0}^{x-1}(x-j)\cdot\{N_j(t_0-x-1)+Z_j(t_0-x-1)-W_j(t_0-x-1)\} + \sum_{k=2}^{x}\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\}\cdot(Z_j(t_0-k)-W_j(t_0-k)) > x\cdot m \tag{21}$$

Multiplying the above equation by $\frac{x+1}{x}$, we get:

$$\sum_{j=0}^{x-1}\frac{x+1}{x}(x-j)\{N_j(t_0-x-1)+Z_j(t_0-x-1)-W_j(t_0-x-1)\} + \sum_{k=2}^{x}\sum_{j=0}^{k-2}\frac{x+1}{x}\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\}(Z_j(t_0-k)-W_j(t_0-k)) > (x+1)\cdot m \tag{22}$$

Using the above equation, we will now prove that $[A'_{x+1}]$ holds in this case. First, observe that

$$(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p \le (k-j-1)+x-k+1 = x-j \le x \quad \text{, for } x>0, j\ge 0.$$

Combining this with the assumption of (Case 1), i.e., $\mathbf{1}_{t_0-x-1}=1$, we get:

$$\begin{aligned}&\frac{x+1}{x}\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\}\\ &=\frac{1}{x}\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\}+\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\}\\ &\le 1+\{(k-j-1)+\sum_{p=t_0-x}^{t_0-k}\mathbf{1}_p\} = (k-j-1)+\sum_{p=t_0-x-1}^{t_0-k}\mathbf{1}_p\end{aligned} \tag{23}$$

Second, to transform the current coefficient ($\frac{x+1}{x}(x-j)$) of $N_j(t_0-x-1)$ to the coefficient of $N_j(t_0-x-1)$ in $[A'_{x+1}]$, we apply the following inequality:

$$\frac{x+1}{x}(x-j) = x+1-\frac{x+1}{x}\cdot j < (x+1-j), \text{ for } x>0, j\ge 0 \tag{24}$$

Now, by applying Eqs. (23) and (24) to Eq. (22), the following inequality holds:

$$\begin{aligned}&\sum_{j=0}^{x-1}(x+1-j)\cdot\{N_j(t_0-x-1)+Z_j(t_0-x-1)-W_j(t_0-x-1)\}\\ &+\sum_{k=2}^{x}\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x-1}^{t_0-k}\mathbf{1}_p\}\cdot(Z_j(t_0-k)-W_j(t_0-k))\\ &=\sum_{j=0}^{x-1}(x+1-j)\cdot N_j(t_0-x-1)\\ &+\sum_{j=0}^{x-1}(x+1-j)\cdot\{Z_j(t_0-x-1)-W_j(t_0-x-1)\}\\ &+\sum_{k=2}^{x}\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x-1}^{t_0-k}\mathbf{1}_p\}\cdot(Z_j(t_0-k)-W_j(t_0-k))\\ &> (x+1)\cdot m\end{aligned} \tag{25}$$

We will arrange terms such that the LHS of Eq. (25) is equal to that of $[A_{x+1}]$. To do this, we use that the following equation which holds for $k=x+1$.

$$\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x-1}^{t_0-k}\mathbf{1}_p\} = \sum_{j=0}^{x-1}\{(x-j)+\mathbf{1}_{t_0-x-1}\} = \sum_{j=0}^{x-1}(x+1-j) \tag{26}$$

Then, the LHS of Eq. (25) can be represented as follows:

$$\sum_{j=0}^{x-1}(x+1-j)\cdot N_j(t_0-x-1) + \sum_{k=2}^{x+1}\sum_{j=0}^{k-2}\{(k-j-1)+\sum_{p=t_0-x-1}^{t_0-k}\mathbf{1}_p\}\cdot(Z_j(t_0-k)-W_j(t_0-k)) \tag{27}$$

We can now check that $[A'_{x+1}]$ holds by adding the non-negative term for $j=x$ to the first summation of the above inequality. Finally, we conclude that if $[A'_x]$ is true, then $[A'_{x+1}]$ is true for this case.

(Case 2) Assume

$$\mathbf{1}_{t_0-x-1}=0 \iff \sum_{j=0}^{x}N_j(t_0-x-1) > m. \tag{28}$$

Since only m tasks can be serviced in $[t_0-x-1, t_0-x)$, there exists a minimum number y ($\le x$) such that at least one of the tasks in $S_y(t_0-x-1)$ is not serviced in $[t_0-x-1, t_0-x)$. It holds that $y\ge 1$ because otherwise $N_0(t_0-x-1)>m$, which means t_0-x is the first instant when there is a task with negative laxity. Thus, all tasks in $S_j(t_0-x-1)$ for $j=0,...,y-1$ are serviced while all tasks in $S_j(t_0-x-1)$ for $j=y+1,...,x$ are not serviced. Among tasks in $S_y(t_0-x-1)$, $\sum_{j=0}^{y}N_j(t_0-x-1)-m$ tasks are not serviced and $m-\sum_{j=0}^{y-1}N_j(t_0-x-1)$ tasks are serviced. Considering that serviced tasks keep their laxity and non-serviced ones reduce their laxity by one, we establish the following relationship between $N_j(t_0-x)$ and $N_j(t_0-x-1)$:

$N_j(t_0-x)=$

$$\begin{cases} N_j(t_0-x-1)+Z_j(t_0-x-1)-W_j(t_0-x-1), & 0\le j\le y-2 \\ N_j(t_0-x-1)+\{\sum_{k=0}^{y} N_k(t_0-x-1)-m\} & \\ +Z_j(t_0-x-1)-W_j(t_0-x-1), & j=y-1 \\ \{m-\sum_{k=0}^{y-1} N_k(t_0-x-1)+N_{j+1}(t_0-x-1)\} & \\ +Z_j(t_0-x-1)-W_j(t_0-x-1), & j=y \\ N_{j+1}(t_0-x-1)+Z_j(t_0-x-1) & \\ -W_j(t_0-x-1), & y+1\le j\le x-1 \end{cases}$$

Using the above equation in $[A'_x]$, we can easily arrive at $[A'_{x+1}]$ after some mathematical simplications. The detailed derivation is given in our technical report [25].

From (Case 1) and (Case 2), the inductive step is correct, and this concludes the proof. ■

B. Proof of Theorem 2

Proof:

To prove this theorem, we investigate how much a task τ_k contributes to the LHS of $[A_x]$ in Theorem 2 and to the LHS of $[A'_x]$ in Lemma 3. Then, we prove that this contribution to $[A_x]$ is always equal to or larger than that to $[A'_x]$. We denote the LHS of $[A_x]$ by (A), and the LHS of $[A'_x]$ by (B).

Let $t_0-t_{\tau_k}$ denote the release time of the latest job of task τ_k before t_0. Further, let $t_0-t_{\tau_k(q)}$ and $t_0-t'_{\tau_k(q)}$ denote the release and finishing times, respectively, of the q^{th} job of τ_k prior to the job released at $t_0-t_{\tau_k}$ (a larger q means earlier job). We also define $t_0-t_{\tau_k(0)} \overset{\Delta}{=} t_0-t_{\tau_k}$.

Since $Z_\theta(t_0-y)$ is the number of tasks whose jobs are released at t_0-y+1 with a laxity of θ, τ_k contributes to (B) through $Z_\theta(t_0-y)$-terms only when $\theta=D_k-C_k$ and $t_o-y=t_0-t_{\tau_k(q)}-1$. Similarly, since $W_\theta(t_0-y)$ is the number of tasks whose jobs are finished at t_0-y+1 and have a laxity of θ at t_0-y, τ_k contributes to (B) through $W_\theta(t_0-y)$-terms only when $t_0-y=t_0-t'_{\tau_k(q)}-1$. Both these contributions to (B) occur at all time instants t_0-x such that $x\ge y$. Finally, at any time instant when τ_k is active (i.e., t_0-x such that $t_0-t_{\tau_k(q)}\le t_0-x\le t_0-t'_{\tau_k(q)}+1$), it contributes to (B) through at most one N-term.

We now consider three cases depending on the value of time instant t_0-x to prove this theorem.

(Case 1) t_0-x, where $x=1,2,...,\min(t_{\tau_k},D_k)$.

Since τ_k does not contribute through any Z-terms after $t-t_{\tau_k}$, it only contributes to (B) at t_0-x through at most one N- and one W-terms. Here the contribution through W-term is negative, and that through the N-term is the same as what τ_k contributes through the $\bar{N}$-term to (A). So, the contribution of τ_k to (A) is equal to or larger than that to (B).

(Case 2) t_0-x, where $x=D_k+1,D_k+2,...,t_{\tau_k}$.

Here, τ_k contributes $x-D_k+C_k$ to (A), because the perceived laxity of τ_k when $t<t_0-D_k$ is D_k-C_k.

Similar to (Case 1), τ_k does not contribute through any Z-terms to (B). Further, since $D_k<t_{\tau_k}$ in this case, the last job of τ_k finishes in the interval $[t_0-x,t_0)$, meaning that τ_k contributes through exactly one W-term to (B). If we denote t_0-y as this job finish time and θ as the laxity of the job at t_0-y-1, the contribution of τ_k through W-term to (B) is $-\{y-\theta+\sum_{p=t_0-x}^{t_0-y-1}\mathbf{1}_p\}$. Additionally, τ_k can contribute to (B) at t_0-x through at most one N-term. If we denote θ' as the laxity of τ_k at t_0-x, this N-term contribution is $x-\theta'$.

Now, during $[t_0-x,t_0-y)$, the job is not executed for exactly $\theta'-\theta$ time units, and execution occurs for at most C_k time units. Then $x-y\le\theta'-\theta+C_k$, for all x such that $D_k+1\le x\le t_{\tau_k}$. That is, in particular, $-y-\theta'+\theta\le C_k-t_{\tau_k}$. Then, as shown by the following inequality, τ_k's contribution to (B) is at most its contribution to (A):

$$\begin{aligned} -\{y-\theta+\sum_{p=t_0-x}^{t_0-y-1}\mathbf{1}_p\}+\{x-\theta'\} &= x-y-\theta'+\theta-\sum_{p=t_0-x}^{t_0-y-1}\mathbf{1}_p \\ &\le x+C_k-t_{\tau_k}\le x+C_k-D_k \end{aligned} \tag{29}$$

(Case 3) t_0-x, where $x=t_{\tau_k}+1,t_{\tau_k}+2,...,\infty$.

Similar to (Case 2), task τ_k contributes $x-D_k+C_k$ to (A).

We now compute the contribution of τ_k to (B) for each time instant. For this purpose, we consider two types of time instants: (Case 3-1) time instants between the finishing time of a job of τ_k and the release time of the next job and (Case 3-2) time instants between the release time of a job of τ_k to the finishing time of the job.

(Case 3-1) t_0-x such that $t_0-t'_{\tau_k(q+1)}\le t_0-x\le t_0-t_{\tau_k(q)}-1$, where $q\ge 0$.

In these time instants, τ_k is inactive. So it cannot contribute through N-terms. Further, at any such instant, there is contribution from at least one Z-term ($Z_{D_k-C_k}(t_0-t_{\tau_k(0)}-1)$). Except for this Z-term, whenever there is contribution from the q^{th} job of τ_k through a Z-term, there is also contribution from the $(q-1)^{st}$ job of τ_k through a W-term. Thus, at any time instant t_0-x in this case, τ_k contributions through $y+1$ Z-terms and y W-terms, where $y+1$ denotes the number of jobs of τ_k released in $[t_0-t_{\tau_k(q)},t_0)$. These contributions to (B) can be expressed as follows:

$$\begin{aligned} &\sum_{q=0}^{y}\{t_{\tau_k(q)}-D_k+C_k+\sum_{p=t_0-x}^{t_0-t_{\tau_k(q)}-1}\mathbf{1}_p\} \\ &\qquad\text{(coefficients of } Z\text{-terms)} \\ &-\sum_{q=1}^{y}\{t'_{\tau_k(q)}-\theta_q+\sum_{p=t_0-x}^{t_0-t'_{\tau_k(q)}-1}\mathbf{1}_p\} \\ &\qquad\text{(coefficients of } W\text{-terms)} \\ &=(y+1)\cdot(-D_k+C_k)+\sum_{q=1}^{y}\theta_q+\{t_{\tau_k(0)}+\sum_{p=t_0-x}^{t_0-t_{\tau_k(0)}-1}\mathbf{1}_p\} \\ &\quad+\sum_{q=1}^{y}\{t_{\tau_k(q)}+\sum_{p=t_0-x}^{t_0-t_{\tau_k(q)}-1}\mathbf{1}_p-t'_{\tau_k(q)}-\sum_{p=t_0-x}^{t_0-t'_{\tau_k(q)}-1}\mathbf{1}_p\} \end{aligned} \tag{30}$$

Using the fact that laxity θ_q cannot be larger than $D_k - C_k$ and re-arranging, it can be easily shown that Eq. (30) is upper-bounded by $-D_k + C_k + x$. Detailed derivation is given in our technical report [25]. Thus, contribution of τ_k to (B) is at most its contribution to (A) under this case.

(Case 3-2) $t_0 - x$ such that $t_0 - t_{\tau_k(q)} \le t_0 - x \le t_0 - t'_{\tau_k(q)} - 1$, where $q \ge 0$.

At these time instants, τ_k contributes through the same number of Z- and W-terms, and at most one N-term from its earliest job. Excluding the contribution through this N-term and the earliest job's W-term, contribution through all other Z- and W-terms is the same as in Eq. (30) of (Case 3-1).

We now calculate the contributions through the earliest job's W- and N-terms. Denote the earliest job's finishing time as $t_0 - t'_{\tau_k(y+1)}$. At $t_0 - x$, contribution of this job through the W-term is $-\{t_{\tau_k(y+1)} - \theta_{y+1} + \sum_{p=t_0-x}^{t_0 - t_{\tau'_k(y+1)}-1}\}$, where θ_{y+1} denotes the laxity of the job at $t_0 - t'_{\tau_k(y+1)} - 1$. Contribution through the N-term is $x - \theta$, where θ denotes the laxity of τ_k's job at $t_0 - x$. Since the laxity of a job is a non-increasing parameter, it holds that $\theta_{y+1} \le \theta$. Then, the contribution of τ_k through its earliest job's W- N-terms can be calculated as follows:

$$
\begin{aligned}
&-\{t_{\tau_k(y+1)} - \theta_{y+1} + \sum_{p=t_0-x}^{t_0 - t'_{\tau_k(y+1)}-1} \mathbf{1}_p\} + \{x - \theta\} \\
&\quad \le -t_{\tau_k(y+1)} - \sum_{p=t_0-x}^{t_0 - t'_{\tau_k(y+1)}-1} \mathbf{1}_p + x \qquad (31)
\end{aligned}
$$

Finally, adding the above quantity to Eq. (30), gives us the overall contribution of τ_k to (B). It can be easily shown that this sum is upper-bounded by $-D_k + C_k + x$. Detailed derivation is given in our technical report [25]. Thus, the contribution of τ_k to (B) is at most its contribution to (A) even in this case.

Finally, using the results of (Case 1), (Case 2), (Case 3-1) and (Case 3-2), we can conclude that the contribution of any task to (A) is larger than or equal to that of its contribution to (B) at any time instant $t_0 - x$, where $x \ge 1$. This concludes the proof. ■

REFERENCES

[1] C. Liu and J. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] J. Leung and J. Whitehead, “On the complexity of fixed-priority scheduling of periodic real-time tasks,” *Performance Evaluation*, vol. 2, pp. 237–250, 1982.

[3] S. Cho, S.-K. Lee, S. Ahn, and K.-J. Lin, “Efficient real-time scheduling algorithms for multiprocessor systems,” *IEICE Trans. on Communications*, vol. E85–B, no. 12, pp. 2859–2867, 2002.

[4] A. Srinivasan and S. Baruah, “Deadline-based scheduling of periodic task systems on multiprocessors,” *Information Processing Letters*, vol. 84, no. 2, pp. 93–98, 2002.

[5] B. Andersson, S. Baruah, and J. Jonsson, “Static-priority scheduling on multiprocessors,” in *RTSS*, 2001, pp. 193–202.

[6] S. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, “Proportionate progress: a notion of fairness in resource allocation,” *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[7] H. Cho, B. Ravindran, and E. D. Jensen, “An optimal real-time scheduling algorithm for multiprocessors,” in *RTSS*, 2006, pp. 101–110.

[8] J. H. Anderson and A. Srinivasan, “Early-release fair scheduling,” in *ECRTS*, 2000, pp. 35–43.

[9] B. Andersson and E. Tovar, “Multiprocessor scheduling with few preemptions,” in *RTCSA*, 2006, pp. 322–334.

[10] K. Funaoka, S. Kato, and N. Yamasaki, “Work-conserving optimal real-time scheduling on multiprocessors,” in *ECRTS*, 2008, pp. 13–22.

[11] B. Andersson and K. Bletsas, “Sporadic multiprocessor scheduling with few preemptions,” in *ECRTS*, 2008, pp. 243–252.

[12] A. Easwaran, I. Shin, and I. Lee, “Optimal virtual cluster-based multiprocessor scheduling,” *Real-Time Systems*, vol. 43, no. 1, pp. 25–59, 2009.

[13] S. K. Lee, “On-line multiprocessor scheduling algorithms for real-time tasks,” in *IEEE Region 10’s Ninth Annual International Conference*, 1994, pp. 607–611.

[14] M. Park, S. Han, H. Kim, S. Cho, and Y. Cho, “Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor,” *IEICE Transaction on Information and Systems*, vol. E88-D, pp. 658–661, 2005.

[15] M. L. Dertouzos and A. K. Mok, “Multiprocessor on-line scheduling of hard-real-time tasks,” *IEEE Transactions on Software Engineering*, vol. 15, pp. 1497–1506, 1989.

[16] T. P. Baker, M. Cirinei, and M. Bertogna, “EDZL scheduling analysis,” *Real-Time Systems*, vol. 40, pp. 264–289, 2008.

[17] J. Y.-T. Leung, “A new algorithm for scheduling periodic, real-time tasks,” *Algorithmica*, vol. 4, pp. 209–219, 1989.

[18] S. Baruah, A. Mok, and L. Rosier, “Preemptively scheduling hard-real-time sporadic tasks on one processor,” in *RTSS*, 1990, pp. 182–190.

[19] M. Bertogna, M. Cirinei, and G. Lipari, “Improved schedulability analysis of EDF on multiprocessor platforms,” in *ECRTS*, 2005, pp. 209–218.

[20] M. Bertogna, M. Cirinei, and G. Lipari, “Schedulability analysis of global scheduling algorithms on multiprocessor platforms,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 553–566, 2009.

[21] M. Bertogna and M. Cirinei, “Response-time analysis for globally scheduled symmetric multiprocessor platforms,” in *RTSS*, 2007, pp. 149–160.

[22] S. Kato and N. Yamasaki, “Global EDF-based scheduling with efficient priority promotion,” in *RTCSA*, 2008, pp. 197–206.

[23] N. Fisher, J. Goossens, and S. Baruah, “Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible,” *Real-Time Systems*, vol. 45, pp. 26–71, 2010.

[24] C. Phillips, C. Stein, E. Torng, and J. Wein, “Optimal time-critical scheduling via resource augmentation,” *Algorithmica*, vol. 32, pp. 163–200, 2002.

[25] J. Lee, A. Easwaran, and I. Shin, “LLF schedulability analysis on multiprocessor platforms,” Dept. of Computer Science, KAIST, South Korea, Tech. Rep. CS-TR-2010-333, 2010.