The Global Limited Preemptive Earliest Deadline First Feasibility of Sporadic Real-time Tasks

Abhilash Thekkilakattil[1], Sanjoy Baruah[2], Radu Dobrin[1], and Sasikumar Punnekkat[1]

[1]Mälardalen University, Sweden
[2]The University of North Carolina at Chapel Hill, USA

***Abstract*—The feasibility of preemptive and non-preemptive scheduling has been well investigated on uniprocessor and multiprocessor platforms under both Fixed Priority Scheduling (FPS) and Earliest Deadline First (EDF) paradigms. While feasibility of limited preemptive scheduling under FPS has been addressed on both uniprocssor and multiprocessor platforms, under EDF it has been investigated only on uniprocessors, and a similar analysis for multiprocessor platforms is still missing.**

In this paper, we introduce global Limited Preemptive Earliest Deadline First (g-LP-EDF) scheduling, and propose the associated feasibility analysis to complete the above described feasibility analysis spectrum. Specifically, we derive a sufficient condition that guarantees g-LP-EDF feasibility of sporadic real-time tasks which directly provides a global Non-Preemptive Earliest Deadline First (g-NP-EDF) feasibility test. We then study the interplay between g-LP-EDF feasibility and processor speed, in order to quantify the sub-optimality of g-NP-EDF in terms of the minimum speed-up required to guarantee g-NP-EDF feasibility of all feasible tasksets. The results presented in this paper complement our previous results on uniprocessors, and provide a unified result on the sub-optimality of non-preemptive EDF on both uniprocessor and multiprocessor platforms.

I. Introduction

The multi-core revolution has lead to a revived interest among researchers and practitioners, particularly in the field of real-time embedded systems, to leverage on the ability of multiprocessing systems to provide higher performance. However the use of multi-core systems in real-time applications requires a careful consideration of the resulting hardware-software ecosystem. For example, preemptively scheduling hard real-time tasks on multi-core platforms typically imply high preemption and migration related overheads (potentially causing deadline misses) resulting from specific hardware features, such as caches, that on the other hand significantly improve average system-performance. Preemption and migration overheads are typically composed of cache related preemption and migration delays (CPMD), pipeline delays and context switch overheads in addition to the increased bus contention costs [1] [2] [3]. The CPMDs are temporal overheads incurred by a preempted task due to loss of cache affinity, that may jeopardize the temporal correctness of the system. Loss of cache affinity also increases bus contention in the system due to frequent accesses to off-chip memory, and translates as temporal overheads [2]. The delay incurred by the scheduler to suspend the currently executing task, save its context, and prepare the execution environment for the preempting task forms the context switch overheads. Preemptions may also require flushing of the instruction pipeline and reloading it with new set of instructions, which also translates as temporal overheads in the schedule. High preemption and migration related overheads are considered to be an emerging problem in many real-time applications [4] [3] [1], e.g., in autonomous vehicles where data intensive operations, such as advanced planning and tracking, form a critical part of the software.

This work was supported by the Swedish Research Council project CONTESSE (2010-4276); NSF grants CNS 1016954, CNS 1115284, and CNS 1218693; and ARO grant W911NF-09-1-0535.

✉[1] firstname.lastname@mdh.se ✉[2] baruah@cs.unc.edu

A major part of real-time scheduling theory deals with fully preemptive scheduling of periodic and sporadic real-time tasks on uniprocessor [5] [6] [7] [8] and multiprocessor [9] [10] [11] [12] platforms. In the worst case, fully preemptive scheduling can lead to prohibitively high preemption and migration related overheads potentially causing deadline misses [2] [13] [14]. This for example happens when 1) there is a large number of preemptions/migrations or 2) these preemptions/migrations occur at points in code where the cost of preemption/migration is significantly high. Bui et. al [15] observed that CPMDs can increase the task execution times by 33% as the overheads can be as high as $655\mu s$ for a single preemption.

Non-preemptive scheduling [4] [16] [17] [18] [19] is often favored in resource constrained applications over preemptive scheduling due to its low runtime overhead, low memory requirements and reduced susceptibility to transient faults [13]. However, the fact that non-preemptive scheduling can be infeasible even at arbitrarily low processor utilization due to blocking on higher priority tasks [20] discouraged extensive adoption of non-preemptive scheduling. Consequently, non-preemptive scheduling has been less investigated when compared to preemptive scheduling since the work by Jeffay et al. [4]. One way to address blocking related infeasibility under non-preemptive scheduling, while also minimizing potentially pernicious preemption costs, is to limit preemptions to selected points in the task code such that the resulting blocking as well as the newly introduced preemption cost will not result in an unschedulable system. Limiting preemptions to pre-determined points in the task code, like non-preemptive scheduling, has the additional advantage of achieving a more simplified and efficient WCET analysis [20]. A number of recent works [21] [22] [23] [24] [25] examined the possibility of limiting preemptions in order to take advantage of the best of preemptive and non-preemptive paradigms. However, all these efforts focused on uniprocessor systems (a detailed survey is available in [26]). Davis et al. [27] recently presented a schedulability analysis technique for the global fixed priority scheduling with deferred preemptions. They showed that multiprocessor fixed priority schedulability can be improved by carefully choosing the task priorities as well as the length of their final non-preemptive regions.

Resource augmentation was first introduced by Kalyanasundaram et al. [28], showing that faster processors can achieve the same effect as clairvoyance. In [29], we derived the resource augmentation bound that guarantees the fault tolerance feasibility of a set of real-time tasks under an error burst of known upper-bounded length. Davis et al. [30] [31] [32] derived the upper and lower bounds on the processor speed-up required for a fixed priority scheduler to schedule all the tasksets scheduled by an optimal scheduling algorithm, leveraging on the optimality of EDF. In their work, the ‘goodness’ or sub-optimality of FPS with respect to an optimal scheduling algorithm such as EDF, is quantified by the processor speed-up required to guarantee the FPS schedulability of the set of all feasible tasks that are schedulable by an optimal algorithm. We later derived the resource augmentation bound that guarantees non-preemptive feasibility of a uniprocessor feasible taskset [33]. We showed that, augmenting the scheduler with a faster processor provides a system designer with the possibility of guaranteeing a *specified limited preemptive behavior* that minimizes the preemption related overheads, and hence broaden the set of schedulable tasksets.

Phillips et al. [12] showed that the resource augmentation bound that guarantees global preemptive EDF (g-P-EDF) feasibility of real-time tasks is equal to $\left(2-\frac{1}{m}\right)$. Baruah et al. [34] presented a test, improving upon [35], to determine the global preemptive EDF feasibility of real-time tasks. This test has a speed-up bound $\left(2-\frac{1}{m}\right)$, and is hence a *speed-up optimal* test (please see [34] for more details). In this paper, we extend Baruah et al.’s work [34] and examine the possibility of limiting preemptions in global EDF. We also study how the preemptive behavior of a taskset changes with respect to the processor speed. The main contributions of this paper are as follows:

1) We first propose a global Limited Preemptive Earliest Deadline First (g-LP-EDF) schedulability analysis for sporadic real-time tasks. Specifically, we derive a sufficient condition that guarantees that every task in the taskset can execute non-preemptively for known upper-bounded length L. This test can then be trivially extended to determine the global Non-Preemptive EDF (g-NP-EDF) feasibility of sporadic real-time tasks.
2) We then quantify the sub-optimality of g-NP-EDF by first examining the consequence of increasing the processor speed on g-LP-EDF feasibility, and then deriving the upper-bound on the speed-up that guarantees a specified limited preemptive as well as non-preemptive behavior of the taskset.

The rest of the paper is organized as follows: We present the system model in Section II and derive the conditions for g-LP-EDF feasibility in Section III, followed by the resource augmentation bounds in Section IV. We then derive tighter bounds of the speed-up factors in Section V before concluding in Section VI.

II. Models and Notations

In this section, we introduce the notations used in the rest of the paper, including the task and processor model and the scheduling model.

A. Task and Processor Model

We consider a set of n sporadic real-time tasks $\Gamma=\{\tau_1, \tau_2, ...\tau_n\}$ executing on m identical processors. Each τ_i is characterized by a minimum inter-arrival time T_i, a worst case execution requirement C_i, and a relative deadline $D_i \leq T_i$. The tasks are ordered according to their increasing deadlines, i.e., $D_i \leq D_{i+1}, 1 \leq i < n$, and D_{min} is used to denote D_1. Similarly, the largest execution time is denoted by $C_{max} = \max_{\forall\tau_i\in\Gamma} C_i$. Without loss of generality, we assume that the default speed of all processors is $s = 1$. In common with [28] [30] [31] [32] [12] [34], we make the simplifying assumption that task execution times scale linearly with the processor speed in order to focus on the theoretical consequences of processor speed-up on the preemptive behavior of real-time tasks under global EDF. In other words we assume that when a processor that is two times faster is used, the worst case execution time becomes $\frac{C_i}{2}$, $\forall\tau_i \in \Gamma$. The model also allows us to use the terms ‘processor speed-up factor’ and ‘processor speed’ interchangeably. Changing the processor speed from $s = 1$ to $s = a$, is equivalent to speeding up the processor by a factor ‘a’.

We assume that each task τ_i can execute non-preemptively for a duration L_i (called its non-preemptive region), and the largest non-preemptive execution of any such task is given by $L = \max_{\forall\tau_i\in\Gamma}(L_i)$. Each of these tasks in Γ generates a sequence of jobs J where a job in J is represented by J_k. The density of a task τ_i is defined as $\delta_i = \frac{C_i}{D_i}$ and the maximum density is defined as

$$\delta_{max} = \max_{\forall\tau_i\in\Gamma}\left\{\frac{C_i}{D_i}\right\}$$

We also define

$$\hat{\delta}_{max} = \max_{\forall \tau_i \in \Gamma} \left\{ \frac{C_i}{D_i - L} \right\}$$

B. Global Limited Preemptive Scheduling Model

We assume a deadline based scheduler: in any time interval $[t_a, t_f)$, first m jobs having the earliest deadlines are assigned to the m processors, with ties broken arbitrarily. Whenever a higher priority job J_i is released and all m processors are busy, with at least one processor executing a lower priority job, all the lower priority jobs begin executing non-preemptively for at most L time units. After at most L time units, the scheduler reschedules the entire set of tasks. In other words, J_i preempts the lowest priority executing job after getting blocked for at most L time units, or is allocated to the first processor that becomes idle. Since after a preemption the task can resume either on the same processor or on a different processor, for convenience, whenever we refer to a preemption we mean preemptions and/or migrations.

C. Forced Forward Demand Bound Function

The *demand* of a sequence of jobs J over a time interval of length t is defined as the cumulative execution time of all the jobs in J scheduled in that interval. The *minimum demand* of a given sequence of jobs generated over an interval of length t is defined as the minimum amount of execution that the sequence of jobs could require within t in order to meet all its deadlines. This concept has been extended to sporadic task systems, where a task τ_i's *maxmin* demand over an interval of length t is defined as the largest minimum demand by any sequence of jobs that could be legally generated by τ_i in t [34].

Baruah et al. [34] introduced the forced forward demand bound function (FF-DBF) that generalized the above concepts on a set of speed-σ processors. The FF-DBF of any task τ_i over a time interval of length t is defined as [34]:

$$\text{FF-DBF}(\tau_i, t, \sigma) = q_i C_i + \begin{cases} C_i & if\ r_i \geq D_i \\ C_i - (D_i - r_i)\sigma & if D_i > r_i \geq D_i - \frac{C_i}{\sigma} \\ 0 & otherwise \end{cases}$$

where, σ is a positive real-number and,

$$q_i \overset{def}{=} \left\lfloor \frac{t}{T_i} \right\rfloor$$

and

$$r_i \overset{def}{=} t \ mod \ T_i$$

The FF-DBF of the taskset, denoted by FF-DBF(Γ, t, σ) is given by,

$$\text{FF-DBF}(\Gamma, t, \sigma) = \sum_{\forall \tau_i} \text{FF-DBF}(\tau_i, t, \sigma)$$

Consequently, the FF-DBF(τ_i, t, σ) can be seen as the *maxmin* demand of τ_i over an interval of length t, where the execution outside the interval occurs on a speed-σ processor.

III. The Global LP-EDF Feasibility of Sporadic Real-time Tasks

We follow a method similar to the one used by Baruah et al. in [34]. Let A denote a work conserving limited preemptive algorithm that misses a deadline while scheduling some legal collection of jobs generated by the taskset Γ. We derive a condition for this to be true by examining A's behavior on some minimal legal collection of jobs generated by Γ on which it misses a deadline. Let t_0 denote the time instant at which a deadline miss occurred due to the limited preemptive execution of some job, and let J_1 be the job that missed the deadline. The arrival time of J_1 is denoted as t_1. Let s denote a constant satisfying $\hat{\delta}_{max} \leq s \leq 1$. Since J_1 misses a deadline due to a blocking of duration no greater than L, it must be the case that J_1 has executed for strictly less than $(t_0 - t_1 - L) \times s$.

Consider a sequence J' of jobs J_i, time instants t_i, and an index q according the following pseudo-code:

```
1 for $i \leftarrow 2, 3, ...$ do
2     Let $J_i$ denote a job that
3     -arrives at some time instant $t_i < t_{i-1}$;
4     -has a deadline after $t_{i-1}$;
5     -has not completed execution by $t_{i-1}$;
6     -has executed for strictly less than $(t_{i-1} - t_i) \times s$
      time units over the interval $[t_i, t_{i-1})$;
7     if there is no such job then
8         $q = (i - 1)$;
9         break out of for loop;
```

We define a subset $J \subset J'$ comprising of k jobs: Let J_k represent the last job among the jobs in J' that is blocked by a lower priority job— so that we can analyze the effect of blocking on the deadline miss at t_0. Then the subset J is composed of the jobs J_i, $i = k, ..., 1$. We then calculate the total executions that occur in the schedule over the interval $[t_k, t_0)$, that lead to a deadline miss at t_0.

Let x_i denote the total length of the time intervals in $[t_i, t_{i-1})$ during which job J_i executes.

Observation III.1.

$$\sum_{i=1}^{k} x_i < (t_0 - t_k - L) \times s$$

Proof: According to our assumption, for each job J_i, $2 \leq i \leq k$,

$$x_i < (t_{i-1} - t_i) \times s$$

Summing up over all $2 \leq i \leq k$, we get:

$$\sum_{i=2}^{k} x_i < (t_1 - t_k) \times s$$

Remember that J_1 has executed for

$$x_1 < (t_0 - t_1 - L) \times s$$

Therefore,

$$\sum_{i=2}^{k} x_i + x_1 < (t_1 - t_k) \times s + (t_0 - t_1 - L) \times s$$

Hence,

$$\sum_{i=1}^{k} x_i < (t_0 - t_k - L) \times s$$

■

Let W denote the amount of execution that occurs in the schedule over the interval $[t_k, t_0)$. In $[t_k, t_0)$ lower priority job executions may interfere with the jobs in J leading to a deadline miss at t_0— we refer to these executions as the *blocking executions*. Let $\hat{W}$ denote the amount of execution that occurs in the schedule over the interval $[t_k, t_0)$, excluding the blocking executions that interfere with J. Similarly, let W_i denote the amount of execution that occurs over the interval $[t_i, t_{i-1})$, excluding the blocking executions that interfere with J, and hence $\hat{W} = \sum_{i=1}^{k} W_i$. Finally, let Y_i denote the amount of blocking executions during the interval $[t_i, t_{i-1})$, that interfere with J, leading to a deadline miss at t_0.

In the following, we derive a lower bound on W, which is the sum of $\hat{W}$ and $\sum_{i=1}^{k} Y_i$.

Observation III.2.

$$W > (m - (m-1) \times s)(t_0 - t_k - L) + mL$$

Proof: We have assumed that J_i has not completed its execution by time instant t_{i-1}. Over $[t_i, t_{i-1})$, all m processors must be executing i) J_i, ii) jobs having higher priority than job J_i or iii) blocking executions from lower priority tasks. Therefore, whenever J_i or blocking executions are not using the processors, all m processors must be executing higher priority jobs.

The lower bound for W_i, $2 \le i \le k$, is given by:

$$W_i \ge m(t_{i-1} - t_i - x_i) + x_i - Y_i$$

Therefore, the duration for which higher priority jobs execute in $[t_k, t_0)$ is obtained by summing up all W_is:

$$\hat{W} \ge m(t_0 - t_k) - (m-1)\sum_{i=1}^{k} x_i - \sum_{i=1}^{k} Y_i$$

Adding and subtracting mL to the right hand side of the equation, we get:

$$\hat{W} \ge m(t_0 - t_k) - mL + mL - (m-1)\sum_{i=1}^{k} x_i - \sum_{i=1}^{k} Y_i$$

Which gives:

$$\hat{W} \ge m(t_0 - t_k - L) - (m-1)\sum_{i=1}^{k} x_i + mL - \sum_{i=1}^{k} Y_i$$

Substituting for $\sum_{i=1}^{k} x_i$ from Observation III.1, we obtain:

$$\hat{W} > m(t_0 - t_k - L) - (m-1)(t_0 - t_k - L) \times s + mL - \sum_{i=1}^{k} Y_i$$

Which gives

$$\hat{W} > (m - (m-1) \times s)(t_0 - t_k - L) + mL - \sum_{i=1}^{k} Y_i$$

We get W by adding $\hat{W}$ and $\sum_{i=1}^{k} Y_i$:

$$W > (m - (m-1) \times s)(t_0 - t_k - L) + mL - \sum_{i=1}^{k} Y_i + \sum_{i=1}^{k} Y_i$$

$$\Rightarrow W > (m - (m-1) \times s)(t_0 - t_k - L) + mL$$

■

In the following, we derive an upper-bound on Y_i that denotes the amount of blocking executions happening during the interval $[t_i, t_{i-1})$ interfering with job J_1, leading to its deadline miss.

Observation III.3.

$$\sum_{i=1}^{k} Y_i \le mL$$

Proof: According to our assumption J_k is the only job in J that is blocked. This means that when job J_k is released (at time instant t_k), $p \ge 1$ processors are executing lower priority jobs. Therefore, the total blocking executions happening in the interval $[t_k, t_0)$, over all m processors, is pL.

When all m processors are executing lower priority jobs at the release time of J_k, all these jobs execute non-preemptively for a duration L, and no more blocking executions occur after this (remember our assumption that J_k is the only job in J that is blocked). Therefore, the maximum blocking executions happening in the interval $[t_k, t_0)$ over all m processors is at most mL. ■

We now find an upper-bound on W in the following observation.

Observation III.4. *If the work-conserving algorithm is g-LP-EDF, then,*

$$W \le \textit{FF-DBF}(\tau, (t_0 - t_k), s) + mL$$

Proof: We are analyzing a minimal unschedulable collection [34] of jobs J, where a deadline miss occurred due to a limited preemptive execution of at most L time units. Since the algorithm is g-LP-EDF, and the jobs in J do not block each other, they execute according to their deadlines. All the jobs in J (i.e., the jobs that are released in $[t_k, t_0)$) have their deadlines within the interval $[t_k, t_0)$.

The amount of execution that jobs of any task τ_i contribute to W is bounded from above by the scenario in which a job of τ_i has its deadline at the end of the interval t_0, and the prior jobs arrive as early as possible. In this scenario, the jobs of τ_i that contribute to W include:

1) at least $q_i \overset{def}{=} \lfloor \frac{(t_0 - t_k)}{T_i} \rfloor$ jobs of τ_i with releases and deadlines entirely in $[t_k, t_0)$, and
2) (perhaps) an additional job that has its deadline at time instant $t_k + r_i$, where $r_i \overset{def}{=} (t_0 - t_k) \bmod T_i$

In the case 2 above, there are two sub-cases:

2.a $r_i \geq D_i$: In this case, the additional job with deadline at $t_k + r_i$ arrives at or after t_k and hence its contribution is C_i.

2.b $r_i < D_i$: In this case, the additional job with deadline at $t_k + r_i$ arrives prior to t_k. Hence this job should have completed at least $(D_i - r_i) \times s$ units of execution prior to time instant t_k. Therefore, the contribution of this job to W is at most $max(0, (D_i - r_i) \times s)$.

From 1 and 2 above, we can conclude that the total contribution of τ_i to W is upper-bounded by FF-DBF$(\tau_i, t_0 - t_k, s)$. Summing up the total contributions of all such $\tau_i \in \Gamma$, we get,

$$\sum_{\forall \tau_i \in \Gamma} \text{FF-DBF}(\tau_i, (t_0 - t_k), s) = \text{FF-DBF}(\Gamma, (t_0 - t_k), s)$$

Finally, according to observation III.3, the total duration for which blocking executions execute in $[t_k, t_0)$ is upper-bounded by mL. Hence,

$$W \leq \text{FF-DBF}(\Gamma, (t_0 - t_k), s) + mL$$

■

Observation III.5. *Suppose that a constrained deadline sporadic task system Γ is not g-LP-EDF feasible upon m unit-speed processors. For each s, $s \geq \hat{\delta}_{max}$, there is an interval of length t such that*

$$\textit{FF-DBF}(\Gamma, t, s) > (m - (m-1)s)(t - L)$$

Proof: The proof follows by chaining the lower bound on W of Observation III.2 with the upper-bound of Observation III.4, where $t = (t_0 - t_k)$. ■

The contrapositive of the Observation III.5 above represents a global limited preemptive EDF schedulability condition.

Theorem III.1. *A taskset Γ, where every $\tau_i \in \Gamma$ executes non-preemptively for a duration of at most L_i time units, is g-LP-EDF feasible if $\exists \sigma : \sigma \geq \hat{\delta}_{max}$ such that $\forall t \geq 0$,*

$$\textit{FF-DBF}(\Gamma, t, \sigma) \leq (m - (m-1)\sigma)(t - L)$$

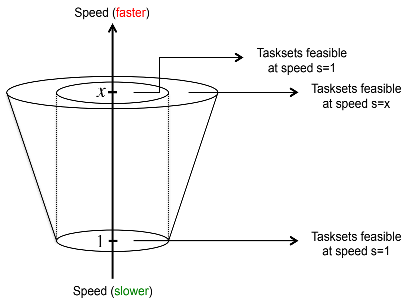
Speed (faster)
Tasksets feasible at speed s=1
x
Tasksets feasible at speed s=x
1
Tasksets feasible at speed s=1
Speed (slower)

Figure 1. Feasibility bucket for tasksets on m processors $(m \geq 1)$

where $L = \max_{\forall \tau_i \in \Gamma}(L_i)$

Proof: The proof follows from the fact that we take the contrapositive of Observation III.5. ■

The intuition behind the above condition is that, in any time interval of length t, the total amount of execution that has to necessarily happen in that interval must be no greater than $t - L$, so that even if the tasks are blocked (for a duration $\leq L$), the tasks complete their execution no later than t. Instantiating the above theorem in the context of a fully non-preemptive scheduler, we get the following result for g-NP-EDF scheduling.

Corollary III.1. *A taskset Γ is g-NP-EDF feasible if $\exists \sigma : \sigma \geq \hat{\delta}_{max}$ such that $\forall t \geq 0$*

$$\textit{FF-DBF}(\Gamma, t, \sigma) \leq (m - (m-1)\sigma)(t - C_{max})$$

where $C_{max} = \max_{\forall \tau_i \in \Gamma}(C_i)$

We have thus obtained sufficient conditions for global limited preemptive and non-preemptive EDF scheduling of sporadic real-time tasks. We can now use an algorithm similar to the one presented in [34] to determine the g-LP-EDF schedulability of a given taskset.

IV. Global LP-EDF Feasibility Vs. Processor Speed

In this paper, one of our aims is to study how the preemptive behavior of real-time tasks change with processor speed under global EDF based scheduling on a multiprocessing platform, which also allows us to quantify the sub-optimality of g-NP-EDF scheduling in terms of bounds on the required speed-up that guarantees g-NP-EDF feasibility. In a previous work [33], we introduced the concept of *feasibility bucket* that illustrated how preemptive behavior of uniprocessor feasible real-time tasksets change with the processor speed. In Figure 1, where we illustrate the concept of feasibility bucket, the base of the bucket represents the set of real-time tasksets feasible on m identical processors $(m \geq 1)$ of speed $s = 1$. On increasing the speed to $s = x$ more tasksets become feasible, and the set of tasksets that were

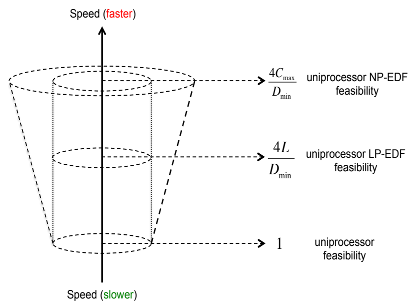
Speed (faster)
$\frac{4C_{max}}{D_{min}}$ uniprocessor NP-EDF feasibility
$\frac{4L}{D_{min}}$ uniprocessor LP-EDF feasibility
1 uniprocessor feasibility
Speed (slower)

Figure 2. The uniprocessor LP-EDF feasibility bucket

originally feasible at speed $s = 1$ becomes a subset of the set of tasksets feasible at speed $s = x$. In Figure 2 we illustrate our previous results, the feasibility bucket for uniprocessor Limited Preemptive EDF (LP-EDF) feasibility of real-time tasks [33]. The base of the bucket represents the uniprocessor feasible tasksets on a processor of speed $s = 1$, and on increasing the processor speed to $\frac{4L}{D_{min}}$, these tasksets become LP-EDF feasible on a uniprocessor. On further increasing the processor speed to $\frac{4C_{max}}{D_{min}}$, they become NP-EDF feasible on a uniprocessor.

In this paper, we extend the uniprocessor LP-EDF feasibility bucket presented in Figure 2 to g-EDF— this is presented in Figure 3. The dotted bucket inside represents the uniprocessor feasibility bucket, and is included to illustrate how the uniprocessor EDF results extend to multiprocessor g-EDF. We refer to this figure as the feasibility bucket for global-LP-EDF scheduling of real-time tasks. The base of the outer bucket denotes the set of all tasksets feasible on m-processors of speed $s = 1$, and are henceforth referred to as *m-processor feasible tasksets*. When the speed of all m processors is increased to $\left(2 - \frac{1}{m}\right)$, all m-processor feasible tasksets are guaranteed g-P-EDF feasibility (as shown in [12]). We show that if the speed of all m processors is further increased to $\left(2 - \frac{1}{m}\right)\frac{4L}{D_{min}}$, all m-processor feasible tasksets are guaranteed g-LP-EDF feasibility such that every task executes non-preemptively for a duration no more than L. We also show that on increasing the speed of all m processors to $\left(2 - \frac{1}{m}\right)\frac{4C_{max}}{D_{min}}$, which corresponds to the height of the outer bucket, all m-processor feasible tasksets are guaranteed g-NP-EDF feasibility (on m-processors).

We now recall the following result from Baruah et al. [34], which is a sufficient condition for global preemptive EDF feasibility.

Theorem IV.1. *A taskset* Γ *is g-P-EDF feasible if* $\exists\sigma : \sigma \geq \delta_{max}$ *such that* $\forall t \geq 0$,

$$\textit{FF-DBF}(\Gamma, t, \sigma) \leq (m - (m-1)\sigma)t$$

We assume that the taskset is initially g-P-EDF feasible and then use the above result by Baruah et al. [34] to derive the following bound on the required speed-up that

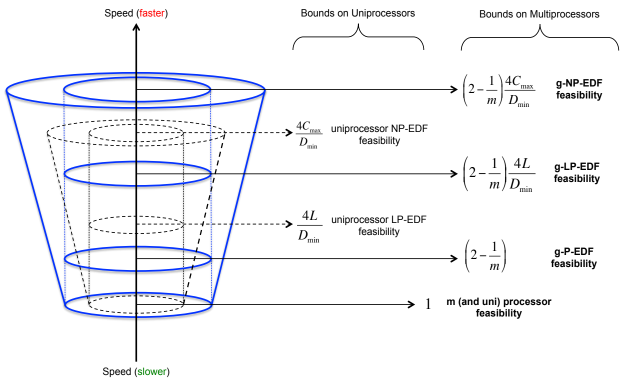
Bounds on Uniprocessors
Bounds on Multiprocessors
Speed (faster)
$\left(2-\frac{1}{m}\right)\frac{4C_{max}}{D_{min}}$ g-NP-EDF feasibility
$\frac{4C_{max}}{D_{min}}$ uniprocessor NP-EDF feasibility
$\left(2-\frac{1}{m}\right)\frac{4L}{D_{min}}$ g-LP-EDF feasibility
$\frac{4L}{D_{min}}$ uniprocessor LP-EDF feasibility
$\left(2-\frac{1}{m}\right)$ g-P-EDF feasibility
1 m (and uni) processor feasibility
Speed (slower)

Figure 3. The multiprocessor g-LP-EDF feasibility bucket

guarantees g-LP-EDF feasibility. We first assume that the condition in Theorem III.1 is not satisfied, and then calculate the processor speed-up required to satisfy the condition in Theorem III.1.

Lemma IV.1. *The speed s that guarantees g-LP-EDF feasibility of any g-P-EDF feasible taskset* Γ*, such that every* $\tau_i \in \Gamma$ *can execute non-preemptively for a duration of at most L, is given by*

$$s \leq \left(\frac{x}{x-1}\right)$$

where, $x = \frac{t}{L}$, $\forall t > 0$.

Proof: According to Theorem III.1, if a taskset is g-LP-EDF feasible,

$$\exists\sigma : \sigma \geq \hat{\delta}_{max}$$

such that $\forall t \geq 0$ and $L = \max_{\forall\tau_i\in\Gamma}(L_i)$,

$$\text{FF-DBF}(\Gamma, t, \sigma) \leq (m - (m-1)\sigma)(t - L)$$

Suppose this is not true, i.e., $\forall\sigma, \forall t \geq 0$,

$$\text{FF-DBF}(\Gamma, t, \sigma) > (m - (m-1)\sigma)(t - L)$$

Since the taskset is g-P-EDF feasible, according to Theorem IV.1,

$$\exists\sigma : \sigma \geq \delta_{max}, \forall t \geq 0$$
$$\text{FF-DBF}(\Gamma, t, \sigma) \leq (m - (m-1)\sigma)t$$

To achieve g-LP-EDF feasibility we can speed-up all m processors such that, for the value of σ for which the Γ is g-P-EDF feasible, the following holds true:

$$\forall t \geq 0 : \frac{\text{FF-DBF}(\Gamma, t, \sigma)}{s} \leq (m - (m-1)\sigma)(t - L)$$
$$\Rightarrow \text{FF-DBF}(\Gamma, t, \sigma) \leq s\left((m - (m-1)\sigma)(t - L)\right)$$

This means that the following condition should be satisfied (since the taskset is both g-LP-EDF and g-P-EDF feasible).

$$s[m - (m-1)\sigma](t - L) \leq [m - (m-1)\sigma]t \Rightarrow s \leq \frac{t}{t - L}$$

Substituting for $x = \frac{t}{L}$, we get,

$$s \leq \frac{x}{x-1}$$

■

We assumed that the taskset is m-processor feasible (and also g-P-EDF feasible), and hence the maximum execution requirement that has to be completely scheduled in any interval of length t is at most t, over all processors. Informally speaking, the above lemma indicates that it is sufficient to speed-up all the processors such that the execution requirement of length t finishes executing in no greater than $t - L$ time units to guarantee g-LP-EDF feasibility (so that lower priority jobs can utilize this slack of length L to execute non-preemptively for at most L units).

Lemma IV.2. *The speed s that guarantees g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute non-preemptively for a duration of at most L, is given by*

$$s \leq \left(2 - \frac{1}{m}\right)\left(\frac{x}{x-1}\right)$$

where, $x = \frac{t}{L}$, $\forall t > 0$.

Proof: According to Phillips et al. [12], the speed-up required to guarantee g-P-EDF feasibility of all m-processor feasible tasksets Γ is upper-bounded by $\left(2 - \frac{1}{m}\right)$. Lemma IV.1 gives the processor speed-up bound that guarantees g-LP-EDF feasibility of a g-P-EDF feasible taskset. Therefore, the speed-up bound that guarantees g-LP-EDF feasibility of all m-processor feasible tasksets is obtained by multiplying the respective bounds. ■

We now derive the speed-up bound when $\frac{t}{L}$ takes different values, e.g., in the following lemma we derive the speed-up bound when $\frac{t}{L} \geq 2$.

Lemma IV.3. *The speed s that guarantees g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration of at most L, where $\frac{t}{L} \geq 2$ and $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times 2$$

Proof: Evaluating the limit of the equation in lemma IV.2 at $x = 2$, we get

$$s = \left(2 - \frac{1}{m}\right) \times 2$$

Evaluating the limit using l'Hopital's rule as x tends to infinity (∞), we get

$$s = \left(2 - \frac{1}{m}\right)$$

Therefore, for any value of $x \in [2, \infty]$,

$$s \leq \left(2 - \frac{1}{m}\right) \times 2$$

■

Lemma IV.4. *The speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration of at most L, where $1 \leq \frac{t}{L} < 2$ and $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times 4$$

Proof: When $1 \leq \frac{t}{L} < 2$, for any $t > 0$, we have $t \geq L$ and $t < 2L$. Since the taskset is m-processor feasible, in the worst case, the maximum execution requirement that has to execute during any time interval of length t, over all processors, is equal to t. Therefore, to guarantee limited preemptive feasibility even under a blocking of at most L units, it is sufficient to speed-up all the processors such that execution requirement t in every time interval finish in $t - L$ time units. Informally speaking, we need to speed-up all m-processors to create a slack of length L in every time interval, which can be utilized by lower priority jobs to finish executing their non-preemptive regions without causing deadline misses.

Let us increase the processor speed by a factor of 2. We will effectively have $t' = 2t$ clock ticks in the time interval t over all processors. Therefore, $\frac{t'}{L} \geq 2$ since $1 \leq \frac{t}{L} < 2$. From lemma IV.3, the speed-up s' required to guarantee g-LP-EDF feasibility for the increased processor speed is

$$s' \leq \left(2 - \frac{1}{m}\right) \times 2$$

As we have already increased the processor speed by a factor of 2, the processor speed that guarantees the g-LP-EDF feasibility of the taskset Γ when $1 \leq \frac{t}{L} < 2$ is

$$s \leq \left(2 - \frac{1}{m}\right) \times 4$$

■

Lemma IV.5. *The speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration of at most L, where $0 < \frac{t}{L} < 1$ and $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

Proof: On increasing the processor speed to $s = \frac{L}{t}$, when $t < L$, the number of available clock ticks in the time interval t increases from t to $t' = t \times \frac{L}{t} = L$, and thus $\frac{t'}{L} = 1$. This is a special case of lemma IV.4, and hence

$$s' \leq \left(2 - \frac{1}{m}\right) \times 4$$

Since we had already increased the processor speed by $\frac{L}{t}$, the speed s that guarantees g-LP-EDF feasibility is

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

■

We obtain the following theorem by combining the speed-up bounds for the three cases presented above.

Theorem IV.2. *The speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration L in any time interval $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

Proof: In the general case, L is bounded by the maximum of the execution times of the tasks in the taskset (i.e., for their fully non-preemptive execution), and t by the shortest deadline. Consequently, $\forall \tau_i$, $\frac{t}{L} > 0$. It follows from Lemmas IV.3 IV.4 IV.5 that the speed-up required to guarantee g-LP-EDF feasibility in the general case is

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

When $\frac{t}{L} \geq 2$, we obtain

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4}{\frac{t}{L}} = \left(2 - \frac{1}{m}\right) \times \frac{4}{2} = \left(2 - \frac{1}{m}\right) \times 2$$

Similarly, when $1 \leq \frac{t}{L} < 2$, we obtain

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4}{\frac{t}{L}} = \left(2 - \frac{1}{m}\right) \times \frac{4}{1} = \left(2 - \frac{1}{m}\right) \times 4$$

Finally when $0 < \frac{t}{L} < 1$, we have

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

Therefore, for any $\frac{t}{L} > 0$, the speed-up required that guarantees g-LP-EDF feasibility is

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

■

Corollary IV.1. *The speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible taskset Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration L, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{D_{min}}$$

This is straightforward, as the value of t that maximizes $\left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$ is the smallest value of t at which the condition in Theorem III.1 should be evaluated, and is given by $t = D_1$ (D_{min}) [34]. We obtain the sub-optimality of g-NP-EDF by substituting $L = C_{max}$, and is formally presented in the following.

Corollary IV.2. *The speed s that guarantees the g-NP-EDF feasibility of an m-processor feasible taskset Γ is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4C_{max}}{D_{min}}$$

We hence obtain the resource augmentation bound, specifically an upper-bound on the required processor speed-up, that guarantees g-LP-EDF feasibility.

V. Improved Resource Augmentation Bounds

In section IV, we presented a unified result on how the preemptive behavior of a set of real-time tasks changes with the processor speed on a multiprocessor system under global EDF based scheduling. In this section, we derive tighter speed-up bounds for the cases in Lemmas IV.4 and IV.5. We know from Lemmas IV.4 and IV.5 that:

1) If $1 \leq \frac{t}{L} < 2$, then $s \leq \left(2 - \frac{1}{m}\right) \times 4$
2) If $0 < \frac{t}{L} < 1$, then $s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$

In the proof for case 1 above (i.e., Lemma IV.4), we applied the bound derived in lemma IV.3 to obtain the result. Similarly, we applied the bound derived in Lemma IV.4 to derive the bound for case 2 above (i.e., Lemma IV.5). However, if we examine the amount of execution requirement that can be additionally executed per unit speed-up, a tighter bound can be obtained for the two cases enumerated above, as detailed in Observations V.1 and V.2.

Observation V.1. *A tighter bound on the speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible task set Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration of at most L, where $1 \leq \frac{t}{L} < 2$ and $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times 2$$

Proof: According to our assumption the task set is m-processor feasible. Therefore, in any time interval of length t, the maximum execution requirement that has to be completely scheduled is at most t, over all the processors. To guarantee limited preemptive feasibility, we need to ensure that the execution requirement t completes in $t - L$ time units on every processor (so that lower priority jobs can utilize this slack of length L to execute non-preemptively for at most L units).

We know that, $1 \leq \frac{t}{L_i} < 2$ and thus we have $t \geq L$ and $t < 2L$. In the worst case, all m-processors are fully occupied during the interval of length t. Therefore it is not possible to feasibly execute the non-preemptive region of length L. Let us assume an increase in the processor speed by a factor of 2. This implies that within an interval of length t, there are in effect $t' = 2t$ clock ticks. It is clear that

$2t \geq L + t$ since $t \geq L$. Thus, any lower priority job can execute non-preemptively on any processor for a duration L within t without causing deadline misses. Therefore, the speed-up required for this case is exactly upper-bounded by 2.

According to Phillips et al. [12] the upper-bound on the speed-up that guarantees g-P-EDF feasibility of the set of all m-processor feasible task sets is $\left(2 - \frac{1}{m}\right)$. Therefore, to guarantee g-LP-EDF feasibility, when $1 \leq \frac{t}{L_i} < 2$ and $t > 0$, the value of s should be

$$s \leq \left(2 - \frac{1}{m}\right) \times 2$$

■

Observation V.2. *A tighter bound on the speed s that guarantees the g-LP-EDF feasibility of an m-processor feasible task set Γ, such that every $\tau_i \in \Gamma$ can execute limited preemptively for a duration of at most L, where $0 < \frac{t}{L} < 1$ and $t > 0$, is given by*

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{2L}{t}$$

Proof: On increasing the processor speed to $s = \frac{L}{t}$, the number of clock ticks in any time interval t increases from t to $t' = t \times \frac{L}{t} = L$. We can now execute the original execution requirement of length t and $L - t$ units of the non-preemptive region L, using the L clock ticks in the time interval t at speed $S = \frac{L}{t}$. Let the remaining length of the non-preemptive region that cannot be executed without a deadline miss in the interval t, be denoted by $L' = t$. We know that $t < L$, thus, in effect we get $\frac{t'}{L'} = \frac{L}{t} > 1$.

Using lemma IV.3, and observation V.1, the exact upper-bound on the speed (denoted by s'), that guarantees g-LP-EDF feasibility is $s' \leq \left(2 - \frac{1}{m}\right) \times 2$.

Since we had already increased the processor speed by $\frac{L}{t}$, the exact upper-bound on the actual speed s is:

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{2L}{t}$$

■

We however do not integrate these results to our main section for the sake of obtaining a unified result on the effect of processor speed on the preemptive behavior. This is because the bound $s \leq \left(2 - \frac{1}{m}\right) \times \frac{2L}{t}$ does not generalize to the case when $\frac{t}{L} \geq 2$. When $\frac{t}{L} \geq 2$, we get $s \leq \left(2 - \frac{1}{m}\right)$, which contradicts Lemma IV.3. However, they are interesting since they indicate that the actual bound on the speed-up required to guarantee the feasibility of a specified limited preemptive behavior is, in practice, lower than the theoretically derived one.

VI. Conclusions and Future Work

In this paper, we derived a sufficient condition for g-LP-EDF feasibility of sporadic real-time tasks. Our test builds on the well known g-P-EDF feasibility test presented by Baruah et al. [34]. We also demonstrate how g-LP-EDF feasibility changes with respect to processor speed, and use this to quantify the sub-optimality of g-NP-EDF. Remarkably, our results on uniprocessors extend directly to the multiprocessor case. This allows us to present a unified result on how the feasibility of limiting preemptions under EDF based scheduling changes with processor speed on uniprocessor and multiprocessor platforms, enabling us to present a unified result on the sub-optimality of non-preemptive EDF on both the platforms. The results presented in this paper can be applied to account for suspensions in global EDF. Current multiprocessor schedulability analysis techniques, particularly all the global EDF analysis, are unable to account for suspensions [36]. If a taskset is deemed g-LP-EDF feasible by our method, in every time interval there exists a slack of length L on all m processors. In the context of a limited preemptive scheduler, this slack of length L enables the limited preemptive execution of lower priority tasks for a duration of at most L. However, in the context of a fully preemptive scheduler, tasks can utilize this slack to potentially suspend for a duration of no more than L, without causing a deadline miss— a detailed study will be presented in a future work.

This paper opens up a number of additional questions on real-time multiprocessor scheduling. As a future work, we plan to examine if the relationship between speed and preemptive behavior holds for any sustainable scheduler. We also plan to examine the effects of having more processors on the preemptive behavior, and how it influences the speed-up bound. Information about the bound on the number of processors, and bound on the speed-up may allow us to design more efficient scheduling algorithms and derive corresponding schedulability tests.

References

[1] A. Bastoni, B. B. Brandenburg, and J. H. Anderson, “Cache-related preemption and migration delays: Empirical approximation and impact on schedulability,” 2010.

[2] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito, and M. Caccamo, “Preemption points placement for sporadic task sets,” in *The Euromicro Conference on Real-Time Systems*, 2010.

[3] B. B. Brandenburg, “Scheduling and locking in multiprocessor real-time operating systems,” Ph.D. dissertation, The University of North Carolina at Chapel Hill, 2011.

[4] K. Jeffay, D. F. Stanat, and C. U. Martel, “On non-preemptive scheduling of periodic and sporadic tasks,” in *The IEEE International Real-time Systems Symposium*, 1991.

[5] S. Baruah, A. Mok, and L. Rosier, “Preemptively scheduling hard-real-time sporadic tasks on one processor,” in *The IEEE International Real-Time Systems Symposium*, 1990.

[6] N. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, “Hard real-time scheduling: The deadline-monotonic approach,” in *The IEEE Workshop on Real-Time Operating Systems and Software*, 1991.

[7] M. Joseph and P. Pandya, “Finding response times in a real-time system,” *The Computer Journal*, 1986.

[8] S. K. Baruah, L. E. Rosier, and R. R. Howell, “Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor,” *Real-Time Systems*, 1990.

[9] T. Baker, “Multiprocessor edf and deadline monotonic schedulability analysis,” in *The IEEE International Real-Time Systems Symposium*, 2003.

[10] M. Bertogna, M. Cirinei, and G. Lipari, “New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors,” in *Principles of Distributed Systems*, ser. Lecture Notes in Computer Science, 2006.

[11] R. Davis and A. Burns, “Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems,” *Real-Time Systems*, 2011.

[12] C. A. Phillips, C. Stein, E. Torng, and J. Wein, “Optimal time-critical scheduling via resource augmentation (extended abstract),” in *The ACM symposium on Theory of computing*, 1997.

[13] M. Short, “The case for non-preemptive, deadline-driven scheduling in real-time embedded systems,” in *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering*, 2010.

[14] H. Ramaprasad and F. Mueller, “Tightening the bounds on feasible preemptions,” in *The ACM Transactions on Embedded Computing Systems*, 2008.

[15] B. D. Bui, M. Caccamo, L. Sha, and J. Martinez, “Impact of cache partitioning on multi-tasking real time embedded systems,” in *The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2008.

[16] L. George, N. Rivierre, and M. Spuri, “Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling,” INRIA, Research Report, 1996.

[17] L. George, P. Muhlethaler, and N. Rivierre, “Optimality and non-preemptive real-time scheduling revisited,” INRIA, Research Report, 1995.

[18] S. K. Baruah, “The non-preemptive scheduling of periodic tasks upon multiprocessors,” *Real-Time Systems*, 2006.

[19] N. Guan, W. Yi, Z. Gu, and G. Yu, “New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms,” in *The IEEE International Real-time Systems Symposium*, 2008.

[20] G. Yao, G. Buttazzo, and M. Bertogna, “Comparitive evaluation of limited preemptive methods,” in *The International Conference on Emerging Technologies and Factory Automation*, 2010.

[21] M. Saksena and Y. Wang, “Scalable multi-tasking using preemption thresholds,” in *In Digest of Short Papers For Work In Progress Session, The 6th IEEE Real-Time Technology and Application Symposium*, 2000.

[22] S. Baruah, “The limited-preemption uniprocessor scheduling of sporadic task systems,” in *The Euromicro Conference on Real-Time Systems*, 2005.

[23] A. Burns, “Advances in real-time systems.” Prentice-Hall, Inc., 1995, ch. Preemptive priority-based scheduling: an appropriate engineering approach.

[24] G. Yao, G. Buttazzo, and M. Bertogna, “Feasibility analysis under fixed priority scheduling with limited preemptions,” *Real-Time Systems*, 2011.

[25] R. Davis and M. Bertogna, “Optimal fixed priority scheduling with deferred pre-emption,” in *The 33rd Real-Time Systems Symposium*, 2012.

[26] G. Buttazzo, M. Bertogna, and G. Yao, “Limited preemptive scheduling for real-time systems: A survey,” *The IEEE Transactions on Industrial Informatics*, 2012.

[27] R. Davis, A. Burns, J. Marinho, V. Nelis, S. Petters, and M. Bertogna, “Global fixed priority scheduling with deferred pre-emption,” in *The IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2013.

[28] B. Kalyanasundaram and K. Pruhs, “Speed is as powerful as clairvoyance,” *Journal of ACM*, 2000.

[29] A. Thekkilakattil, R. Dobrin, S. Punnekkat, and H. Aysan, “Resource augmentation for fault-tolerance feasibility of real-time tasks under error bursts,” in *The 20th International Conference on Real-Time and Network Systems*, 2012.

[30] R. Davis, T. Rothvoss, S. Baruah, and A. Burns, “Exact quantification of the sub-optimality of uniprocessor fixed priority pre-emptive scheduling,” *Real-Time Systems*, 2009.

[31] R. Davis, L. George, and P. Courbin, “Quantifying the Sub-optimality of Uniprocessor Fixed Priority Non-Pre-emptive Scheduling,” in *18th International Conference on Real-Time and Network Systems*, 2010.

[32] R. I. Davis, T. Rothvoss, S. K. Baruah, and A. Burns, “Quantifying the sub-optimality of uniprocessor fixed priority pre-emptive scheduling for sporadic tasksets with arbitrary deadlines,” in *17th International Conference on Real-Time and Network Systems*, 2009.

[33] A. Thekkilakattil, R. Dobrin, and S. Punnekkat, “Quantifying the sub-optimality of non-preemptive real-time scheduling,” in *The Euromicro Conference on Real-Time Systems*, 2013.

[34] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller, “Improved multiprocessor global schedulability analysis,” *Real-Time Systems*, 2010.

[35] V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller, “A constant-approximate feasibility test for multiprocessor real-time scheduling,” *Algorithmica*, 2012.

[36] B. Brandenburg and J. Anderson, “Optimality results for multiprocessor real-time locking,” in *The IEEE International Real-Time Systems Symposium,*, 2010.

VII. Appendix

This appendix corrects a minor representational error in the paper, specifically in Theorem IV.2. In Theorem IV.2, the speed-up bound in given as,

$$s \leq \left(2 - \frac{1}{m}\right) \times \frac{4L}{t}$$

However, the above equation does not emphasize on the fact that, instead of the value given by $\frac{L}{t}$, we should consider the *limiting values* of $\frac{L}{t}$, given by,

$$\begin{cases} \frac{1}{2} & \text{if } \frac{t}{L} \geq 2 \\ 1 & \text{if } 1 \leq \frac{t}{L} < 2 \\ \frac{L}{t} & \text{if } 0 < \frac{t}{L} < 1 \end{cases}$$

In order to integrate this information to the theorems, we define a function $f(L,t)$ given by,

$$f(L,t) = \begin{cases} \frac{1}{2} & \text{if } \frac{t}{L} \geq 2 \\ 1 & \text{if } 1 \leq \frac{t}{L} < 2 \\ \frac{L}{t} & \text{if } 0 < \frac{t}{L} < 1 \end{cases}$$

The speed-up bound in Theorem IV.2 should be:

$$s \leq \left(2 - \frac{1}{m}\right) \times 4f(L,t)$$

Therefore, the speed-up bound in corollary IV.1 should be:

$$s \leq \left(2 - \frac{1}{m}\right) \times 4f(L, D_{min})$$

Similarly, the speed-up bound in Corollary III.1 should be:

$$s \leq \left(2 - \frac{1}{m}\right) \times 4f(C_{max}, D_{min})$$

Consequently, the revised uniprocessor feasibility bucket is given in Figure 4 and the revised feasibility bucket for global EDF is given in Figure 5.

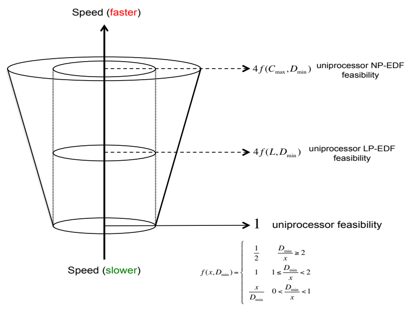
Speed (faster)
$4f(C_{\max}, D_{\min})$ uniprocessor NP-EDF feasibility
$4f(L, D_{\min})$ uniprocessor LP-EDF feasibility
1 uniprocessor feasibility
Speed (slower)
$f(x, D_{\min}) = \begin{cases} \frac{1}{2} & \frac{D_{\min}}{x} \geq 2 \\ 1 & 1 \leq \frac{D_{\min}}{x} < 2 \\ \frac{x}{D_{\min}} & 0 < \frac{D_{\min}}{x} < 1 \end{cases}$

Figure 4. The revised uniprocessor LP-EDF feasibility bucket

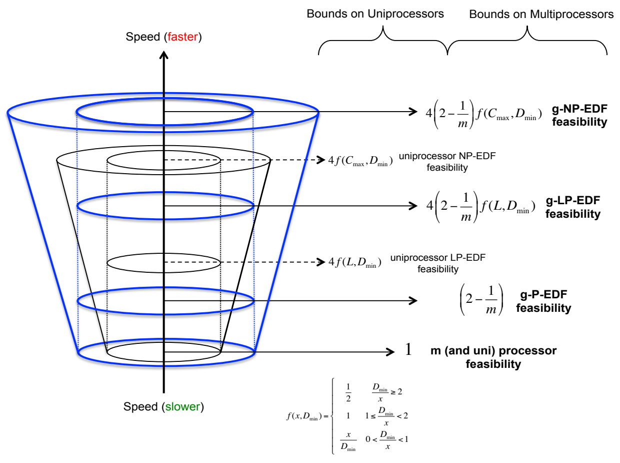
Bounds on Uniprocessors
Bounds on Multiprocessors
Speed (faster)
$4\left(2-\frac{1}{m}\right)f(C_{\max}, D_{\min})$ g-NP-EDF feasibility
$4f(C_{\max}, D_{\min})$ uniprocessor NP-EDF feasibility
$4\left(2-\frac{1}{m}\right)f(L, D_{\min})$ g-LP-EDF feasibility
$4f(L, D_{\min})$ uniprocessor LP-EDF feasibility
$\left(2-\frac{1}{m}\right)$ g-P-EDF feasibility
1 m (and uni) processor feasibility
Speed (slower)
$f(x, D_{\min}) = \begin{cases} \frac{1}{2} & \frac{D_{\min}}{x} \geq 2 \\ 1 & 1 \leq \frac{D_{\min}}{x} < 2 \\ \frac{x}{D_{\min}} & 0 < \frac{D_{\min}}{x} < 1 \end{cases}$

Figure 5. The revised multiprocessor g-LP-EDF feasibility bucket