A note on the sub-optimality of non-preemptive real-time scheduling

Fathi Abugchem, Michael Short, and Donglai Xu

***Abstract*—In this paper, processor speedup analysis is used to strengthen recent results regarding the sub-optimality of uniprocessor non-preemptive Earliest Deadline First (npEDF) scheduling. The sub-optimality of npEDF is defined as the minimum amount of increase in the processor speed that is needed to guarantee the npEDF schedulability of any feasible task set. We show that any preemptively schedulable task set that is not schedulable by npEDF will become schedulable on a processor speeded up by a factor of not more than one plus the value of the largest execution requirement divided by the shortest relative deadline of any task. This reduces the pessimism compared to the best previous bound by factor of at least two. In addition, for the case of non-preemptive Fixed Priority scheduling, we also show that twice this speedup bound is enough to guarantee the schedulability of any feasible task set.**

***Index Terms*—Non-preemptive scheduling, processor speedup factor, resource augmentation, scheduling, sub-optimality.**

I. Introduction

SUB-OPTIMALITY refers to the quantification of the capability of a non-optimal algorithm to successfully schedule *feasible* task sets. A task set is said to be *feasible* if it can be scheduled by an optimal scheduling algorithm. For uniprocessor scheduling, preemptive EDF is known to be optimal while preemptive Fixed Priority (FP) and non-preemptive scheduling schemes are not optimal [1][2]. It was shown in [3] and [4] that non-idling, non-preemptive EDF (npEDF) is optimal among non-preemptive uniprocessor scheduling algorithms for sporadic task systems or periodic task systems without specified start times. This is in the sense that npEDF can schedule any such task set for which a non-idling, non-preemptive schedule exists. If inserted idle-time is allowed and the tasks are periodic with specified start times, the exact scheduling problem is strongly NP-hard [3][4]. In this paper, like previous work, we do not consider the exact analysis of these latter task systems. In previous works resource augmentations, specifically processor speedup measures have been proposed in order to quantify the sub-optimality of non-optimal scheduling algorithms. Quantifying the sub-optimality of FP scheduling was first investigated in [1] and [2], which showed that any *feasible* implicit deadline task set is also schedulable by FP if the speed of the processor is increased by factor of not more than 1.44270. This work was extended for constrained deadline task sets, and a speedup factor not more than 1.76322 was found to be required [5]. For arbitrary deadline task sets, it was shown that 2 is the upper bound on the processor speedup factor needed to guarantee schedulability with FP [6]. Quantifying the sub-optimality of non-preemptive Fixed Priority (npFP) scheduling was investigated in [7]. It was shown that 2 is the upper bound of the processor speedup factor needed to guarantee that any npEDF schedulable task set is also npFP schedulable [7]. Recently, Thekkilakattil et al. [8][9] quantified the sub-optimality of npEDF scheduling compared to EDF; any preemptively schedulable task set is also schedulable by npEDF with a processor speed not more than $(4\,c_{max}/d_{min})$ times faster, where c_{max} represents the largest execution requirement of the task set and d_{min} is the shortest relative deadline. However, it was later shown that this bound does not hold in the general case; a corrected representation was subsequently presented in [10] and given by:

$$S = \begin{cases} 8 & : \quad \frac{d_{min}}{c_{max}} \geq 2 \\ 4 & : \quad 1 \leq \frac{d_{min}}{c_{max}} < 2 \\ 4\frac{c_{max}}{d_{min}} & : \quad 0 < \frac{d_{min}}{c_{max}} < 1 \end{cases}$$

Where S is the bound on the speedup factor. In this paper, a tighter upper bound on the processor speedup factor needed to guarantee npEDF schedulability of any *feasible* task set is introduced. This upper bound is simple in form and valid for periodic and sporadic task sets with arbitrary deadlines. Furthermore we show that this bound - along with the bound presented in [7] - can be used to quantify the sub-optimality of npFP scheduling with respect to EDF, again with a very simple expression.

The remainder of the paper is organized as follows. Section II describes the system model and the key previous results of npEDF schedulability analysis. Section III presents the main contributions of the paper, the processor speedup algorithm and the upper bound of the processor speedup factor for npEDF scheduling. Conclusions are given in section IV.

II. Schedulability Analysis of Non-Preemptive EDF

In this section, the previous work on the non-idling npEDF schedulability analysis of uniprocessor real-time task sets is described. The processor speed is denoted by S and included throughout this analysis; it must be noted that in many previous works, the speed was not explicitly considered as a unit-speed processor was implicitly assumed ($S = 1$).

Manuscript received December 16, 2014; accepted April 7, 2015. Date of publication 00/00/0000; date of current version 00/00/0000. This manuscript was recommended for publication by

The authors are with the Electronics and Control Group, Teesside University, Middlesbrough, UK (e-mail: f.abugchem@tees.ac.uk; m.short@tees.ac.uk; d.xu@tees.ac.uk)

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2015.

A. System Model

It is assumed that the system is implemented on a single-processor platform and the application software consists of a set T of n real-time periodic/sporadic tasks. Each task in this set is parameterized as $\tau_i = (p_i, c_i^S, d_i)$, in which p_i represents the period of periodic tasks (or equivalently, the minimum inter-arrival separation of sporadic tasks), c_i^S represents the worst-case computation requirement of the task when executed on a processor with speed $S>0$, and d_i is the task relative deadline. It is assumed that p_i and d_i are both positive; however there is no restriction on the relation between the period and the relative deadline of each task; the latter may be smaller than, equal to or larger than the former. Note that the task computation time is assumed inversely proportional to the processor speed S, and a linear relationship is assumed. Task periods and deadlines remain unaffected by the processor speed as they are related to an external time reference.

B. Schedulability Analysis

Let the processor utilization of the task set at a processor speed S is defined as $U^s$and given by $U^s = \sum_{i=1}^{n} U_i^s$, where U_i^s represents the utilization of a task τ_i executing on a processor at speed S. Hence $U_i^s = \frac{c_i^s}{p_i}$. Following the worst-case arrival pattern of all tasks at t =0 (synchronous), the worst-case computational demand placed on the CPU by the task set during a time interval $[0, t)$ at a processor speed S can be denoted as $h^s(t)$ and given by [9][11][13]:

$$h^s(t) = \sum_{i=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - d_i}{p_i} \right\rfloor\right\} . c_i^s \quad (1)$$

Let the worst-case blocking due to non-preemption during the time interval $[0, t)$ at a processor speed S be denoted as $b^s(t)$ and given by [12] [13]:

$$b^s(t) = \max_{d_j > t}\{c_j^s\} \quad (2)$$

Based upon results of [8][9][12] a task set with arbitrary deadlines is schedulable under non-idling npEDF at processor speed S if and only if $U^s \leq 1$ and:

$$h^s(t) + b^s(t) \leq t, \ \ \forall t, d_{min} \leq t < L \quad (3)$$

Where L is the end point of a sufficiently long testing interval and is finite when $U^1 \leq 1$ when considering speedup factors $S \geq 1$. d_{min} is the smallest relative deadline among the tasks. Note that the schedulability conditions captured in (1), (2) and (3) are simple extensions of standard, known results to explicitly model the speedup factor S [8-13]. Various methods are known to bound L based upon the parameters of the task set, [13] provides a good discussion. In this paper, an adaptation of the bound derived for preemptive EDF scheduling in [14] is employed, with a trivial extension to include the effects of blocking:

$$L = \max\left\{(d_1 - p_1), \ldots, (d_n - p_n), \frac{c_{max}^s + \sum_{i=1}^{n}(p_i - d_i)U_i^s}{(1 - U^s)}\right\} \quad (4)$$

Although all absolute deadlines in the interval $[d_{min}, L)$ potentially need to be checked, the 'QPA' algorithm described in [14] can be employed to significantly reduce the number of deadlines to be evaluated in the average case.

III. Quantifying the Sub-Optimality of Non-Preemptive EDF Scheduling

In this section, we derive an upper bound on the processor speedup factor required to guarantee npEDF scheduling of unit-speed feasible task sets. This bound is valid for sporadic and periodic task sets with implicit, constrained, and arbitrary deadlines. We start with a general result:

Theorem 1: The processor speed S that guarantees the non-idling npEDF schedulability of any task set is given by:

$$S = \max_{d_{min} \leq t < L}\left\{\frac{h^1(t) + b^1(t)}{t}\right\} \quad (5)$$

Where, $h^1(t)$ and $b^1(t)$ are the processor demand (1) and worst-case blocking (2) at unit processor speed (S =1).

Proof: Let us assume that the task set is initially executing on a processor of unit-speed ($i.e.\ S = 1$) and that the execution requirements of each task in the set T scale linearly with the processor speed. In this case, we have that:

$$c_i^s = \frac{c_i^1}{S}, \qquad \forall i, 0 < i \leq n \quad (6)$$

The task set under npEDF on the processor speed S = 1 is deemed not schedulable if at any time t during a time interval $[d_{min}, L)$ the processor demand function plus worst-case blocking exceeds the value of t:

$$t < h^1(t) + b^1(t) \quad (7)$$

In order to guarantee the schedulability of the task set, the right hand side of (7) should be decreased below t, which can be achieved by increasing the processor speed. From (6) we see that at speed S the task set is schedulable if:

$$t \geq \frac{h^1(t) + b^1(t)}{S} \ , \ \forall t \ , d_{min} \leq t < L \quad (8)$$

Rearranging this condition in terms of S gives:

$$S \geq \frac{h^1(t) + b^1(t)}{t} \ , \ \forall t \ , d_{min} \leq t < L \quad (9)$$

Taking the maximum over all t in the test interval gives the required result:

$$S = \max_{d_{min} \leq t < L}\left\{\frac{h^1(t) + b^1(t)}{t}\right\} \quad \blacksquare$$

Thus, if $S \leq 1$ then the task set is schedulable without pre-emption on a unit-speed processor. A value of $S > 1$ occurs when a deadline is missed (i.e. $h^1(t) + b^1(t) > t$). Therefore the above theorem can be used to state the non-preemptive schedulability of the task set. Consequently, we derive an upper bound on the processor speedup factor required to guarantee the schedulability of a *feasible* task set for npEDF scheduling. A similar approach to that employed in [8][9][10] is taken, however a tighter bound is achieved.

Theorem 2: The processor speedup factor S that is needed to guarantee the schedulability of any *feasible* task set with arbitrary deadlines under npEDF scheduling is upper-bounded by the quantity:

$$S \leq 1 + \frac{c_{max}}{d_{min}} \quad (10)$$

Proof: In order to ensure the non-preemptive schedulability of any unit-speed feasible task set, the speed S must be set such that the slack time $t - h^S(t)$ must be at least as big as the non-preemptive blocking $b^S(t)$ for all values of $t >= d_{min}$. Since

$h^S(t)=h^1(t)/S$ and $b^S(t)=b^1(t)/S$ this can be expressed as the condition below:

$$\forall t\,, d_{min} \le t < L;\;\; t - \frac{h^1(t)}{S} \ge \frac{b^1(t)}{S} \tag{11}$$

Solving for S gives:

$$\forall t\,, d_{min} \le t < L;\; S \ge \frac{h^1(t) + b^1(t)}{t} \tag{12}$$

As the task set is assumed to be feasible, the maximum value of the processor demand at unit-speed $h^1(t)$ is equal to t. In addition, it follows from (2) that the maximum value of $b^1(t)$ is c^1_{max}. Substituting this information into (12):

$$S \ge \frac{t + c^1_{max}}{t} = 1 + \frac{c^1_{max}}{t} \tag{13}$$

Maximizing the right hand side of (13) over t, subject to $t \in [d_{min}, L)$ results in setting t to the shortest task relative deadline d_{min}. Substituting $t = d_{min}$ into (13) gives the upper bound on the required S which completes the proof. ■

The value of the bound on S is plotted in Fig. 1. As seen from equation (10) and Fig. 1, this bound dramatically increases as d_{min} goes below c_{max} and approaches to one as the value of d_{min} becomes greater than c_{max}. If time is restricted to be discrete and task parameters are taken to be integer (as is often the case), then limits in the value of the bound can be obtained independently of d_{min}. From equation (10) the maximum value of S is at the minimum value of d_{min}. Assuming discrete time, the minimum d_{min} is one; the blocking factor can also be reduced to c_{max}-1 [13]. Using this information in (10), we get the upper limit $S \le c_{max}$. For the lower limit on S it is well known that any *feasible* task set having $c_{max} = 1$ can be scheduled non-preemptively without the need to speed up the processor, hence trivially $S \ge 1$.

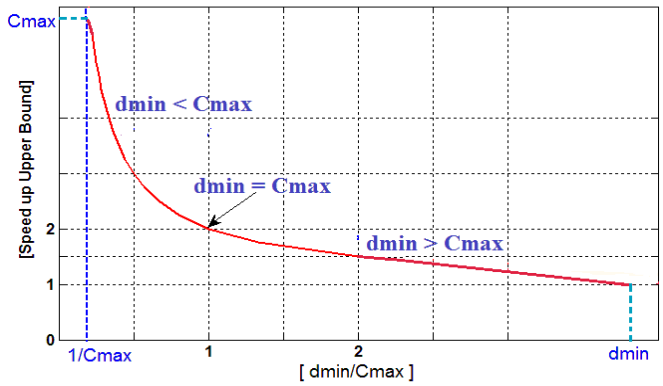

Fig. 1. Upper bound of the processor speedup.

Next, we show that the bound in (10) is tighter than the previous one presented in [8] [9] [10]. We prove this by showing that the limits of this bound is less than the bound presented in [10]. This is done by evaluating the limits of (10) for the same three extreme cases considered in [8], [9] and [10]. Proceeding:

CASE 1: The speed S that guarantees npEDF schedulability of any *feasible* task set is upper-bounded by 1.5 if $\frac{d_{min}}{c^1_{max}} \ge 2$.

Proof: Evaluating the limits of (10) at $d_{min} = 2c^1_{max}$, we get:

$$S \le 1 + \frac{c^1_{max}}{d_{min}} = 1 + \frac{c^1_{max}}{2c^1_{max}} = 1.5 \tag{14}$$

According to the equation (10) and as seen from (14) the value of S decreases as the value of d_{min} increases, i.e. S is less than 1.5 if $d_{min} > 2c^1_{max}$, hence the value of S is bounded by 1.5 if $\frac{d_{min}}{c^1_{max}} \ge 2$. ■

CASE 2: The speed S that guarantees npEDF schedulability of any feasible task set is upper-bounded by 2, if $1 \le \frac{d_{min}}{c^1_{max}} < 2$.

Proof: Evaluating the lower limit of this case, i.e. when $c^1_{max} = d_{min}$, we get:

$$S \le 1 + \frac{c^1_{max}}{d_{min}} = 1 + \frac{c^1_{max}}{c^1_{max}} = 2 \tag{15}$$

Clearly the bound is linearly decreasing for increasing d_{min}. At the upper limit of this case, i.e. when $c^1_{max} = 2d_{min}$, CASE 1 has shown that $S \le 1.5$ Accordingly 2 is a valid upper-bound for S when $1 \le \frac{d_{min}}{c^1_{max}} < 2$. ■

CASE 3: The speed S that guarantees npEDF schedulability of any feasible task set is upper-bounded by $\frac{2\,c^1_{max}}{d_{min}}$, if $0 < \frac{d_{min}}{c^1_{max}} < 1$.

Proof: In this case $d_{min} < c^1_{max}$, and supposing the processor speed has been increased to $S' = \frac{c^1_{max}}{d_{min}}$, then the value of $c^{S'}_{max} = \frac{c^1_{max}}{S'} = d_{min}$. According to CASE 2 the upper bound on the processor speed will then be 2 (as now d_{min}=c_{max}). Since the processor speed has been already increased by $\frac{c^1_{max}}{d_{min}}$, the upper bound on the actual speed S is $\frac{2\,c^1_{max}}{d_{min}}$. ■

Based on these three cases the upper bound of the processor speedup factor can also be presented in the form given in [10]:

$$S = \begin{Bmatrix} 1.5 & : & \frac{d_{min}}{c_{max}} \ge 2 \\ 2 & : & 1 \le \frac{d_{min}}{c_{max}} < 2 \\ 2\frac{c_{max}}{d_{min}} & : & 0 < \frac{d_{min}}{c_{max}} < 1 \end{Bmatrix} \tag{16}$$

Comparing this bound with the previous, one observes that each case is tighter by at least a two-fold factor. Fig. 2 shows a comparison between both bounds as the ratio d_{min}/c_{max} increases. The improvement in the current bound is due to the observation that both the processor demand *and* the worst-case blocking due to non-preemption can be scaled with the processor speed, as when the processor speed increases/decreases, the execution time of all tasks is assumed to decrease/increase in proportion - including that of the task with the index satisfying the worst-case blocking function (2).

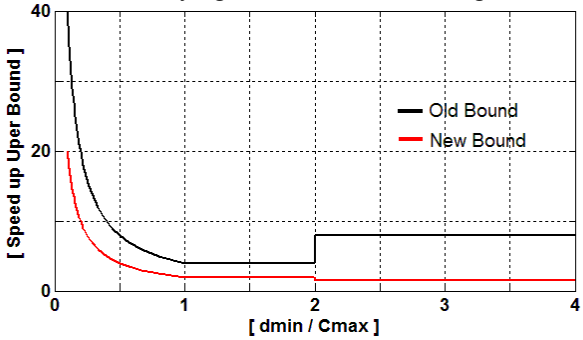

Fig. 2. Previous and new upper-bounds of the processor speedup.

Theorem 2 can also be used to drive a useful upper bound on the required processor speedup if all deadlines of the task set are implicit (i.e. $d_i = p_i \,, \forall i \,, 0 < i \leq n$).

Theorem 3: The minimum processor speedup factor needed to guarantee the schedulability of a *feasible* implicit deadline task set under npEDF scheduling is upper-bounded by:

$$S \leq U^1 + \frac{c^1_{max}}{d_{min}} \tag{17}$$

Proof: Applying the same arguments as in Theorem 2, the speed S which guarantees schedulability satisfies the conditions of (12). For feasible implicit deadline task sets, the maximum value of the processor demand at unit-speed is bounded by the utilization factor, i.e. $h^1(t) \leq U^1 t$. Substituting this in (12) along with $b^1(t) \leq c^1_{max}$:

$$S \geq \frac{U^1.t + c^1_{max}}{t} = U^1 + \frac{c^1_{max}}{t} \tag{18}$$

As before, maximizing the right hand side of (18) over t subject to $t \in [d_{min}, L)$ gives the upper bound on the required speedup S:

$$S \leq U^1 + \frac{c^1_{max}}{d_{min}} \tag{19}$$

■

The effect of utilization in the case of an implicit-deadline task set is illustrated in the above, in that if the unit-speed CPU utilization is lowered without altering the key task parameters c^1_{max} and d^1_{min}, the required speedup factor may be reduced. The above theorems can also be used, along with key previous results, to determine simple upper bounds on the processor speedup factor required for npFP.

Corollary 1: The minimum processor speedup factor that is needed to guarantee the schedulability of a feasible arbitrary deadline task set under npFP with optimal priority assignment is upper-bounded by:

$$S \leq 2 + \frac{2c_{max}}{d_{min}} \tag{20}$$

Proof: It has been shown in [7] that the minimum amount of the processor speedup factor needed to guarantee npFP scheduling (with an optimal priority assignment [15]) of any npEDF schedulable task set is not more than 2. Suppose that f_{np} is defined as the upper bound of the processor speedup of npEDF and is given as in equation (10).

Elaborating in the result of theorem 2, any task set which is schedulable by EDF is also schedulable by npEDF if the processor speed has been increased by f_{np}, and based on [7] any task set schedulable by npEDF is also schedulable by npFP if the processor speed has been increased by 2. Accordingly the upper bound of the processor speedup factor which is needed to guarantee the schedulability of a *feasible* task set under npFP is given as:

$$S = 2\, f_{np} \tag{21}$$

Substituting the value of f_{np} from equation (10) in the equation above completes the proof. ■

IV. Conclusion

In this paper, resource augmentation measures have been used to evaluate the upper bound on the required processor speedup factor needed to guarantee npEDF scheduling of arbitrary deadline *feasible* task sets. We have shown that the processor speedup factor is not more than $1 + \frac{c_{max}}{d_{min}}$. It has been proven that this bound is tighter than the previous one which is presented in [10] and also has a simpler form. For implicit deadline task sets a potentially tighter bound was also obtained. We also derived a simple but useful expression which quantifies the sub-optimality of npFP scheduling for *feasible* task sets having an optimal priority assignment.

References

[1] C.L Liu and J. W. Layland, ‘‘Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment,’’ J. ACM, vol. 20, no. 1, pp. 40-61, 1973.

[2] M.L. Dertouzos ,‘‘Control Robotics: The Procedural Control of Physical Processes,’’ Proc. Int Federation for Information Processing (IFIP) Congress, pp. 807-813, 1974.

[3] K. Jeffay, D.F. Stanat and C.U. Martel, ‘‘On Non-Preemptive Scheduling of Periodic and Sporadic Tasks,’’ In Proceedings of the IEEE Real-Time Systems Symposium, pp. 129-139, 1991.

[4] L. George, P. Muhlethaler, N. Rivierre, ‘‘Optimality and Non-Preemptive Real-Time Scheduling Revisited’’ Rapport de Recherche RR-2516, INRIA, Le Chesnay Cedex, France, 1995.

[5] R. I. Davis, T. Rothvoß, S. K. Baruah, A. Burns, ‘‘Exact Quantification of the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling.’’ Real-Time Systems, vol. 43, no. 3, pp. 211-258, 2009.

[6] R. I. Davis, T. Rothvoß , S.K. Baruah, A. Burns, ‘‘Quantifying the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling for Sporadic Task sets with Arbitrary Deadlines,’’ In proceedings of Real-Time and Network Systems (RTNS'09), pp. 23-31, 2009.

[7] R. Davis, L. George, P. Courbin, ‘‘Quantifying the Suboptimality of Uniprocessor Fixed Priority Non-Pre-emptive Scheduling,’’ In 18th International Conference on Real-Time and Network Systems, 2010.

[8] A. Thekkilakattil, “Resource Augmentation for Performance Guarantees in Embedded Real-Time Systems,” PhD Thesis School of Innovation Design and Engineering, Malardalen University Sweden, 2012.

[9] A. Thekkilakattil, R. Dobrin, S. Punnekkat, “Quantifying the Sub-optimality of Non-preemptive Real-time Scheduling,’’ In: Proc. of 25th Euromicro Conference on Real-Time Systems, Paris, France, 2013.

[10] A.Thekkilakattil, S. Baruah, R. Dobrin, and S. Punnekkat, “The Global Limited Preemptive Earliest Deadline First Feasibility of Sporadic Real-time Tasks”, In: Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS), Madrid, Spain, 8-11 July 2014.

[11] S. K. Baruah, A. K. Mok, L. E. Rosier, “Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor,’’ In Proc. RTSS, pp. 182-190, 1990.

[12] L. George, N. Rivierre, M. Spuri, “Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling,’’ INRIA Research Report, No. 2966, September 1996.

[13] J. A. Stankovic, M. Spuri, K. Ramamritham, G. C. Buttazzo, “Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms,’’ Kluwer Academic Publishing, 1998.

[14] F. Zhang, A. Burns, “Schedulability Analysis for Real-Time Systems with EDF Scheduling,’’ IEEE Transactions on Computers, Vol. 58, No. 9, pp. 1250-1258, 2009.

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