Scheduling of mixed-criticality sporadic task systems with multiple levels

Sanjoy K. Baruah * Vincenzo Bonifaci †
Gianlorenzo D'Angelo (Speaker) ‡ Haohan Li §
Alberto Marchetti-Spaccamela ¶ Suzanne van der Ster ‖ Leen Stougie ‖

There is an increasing trend in embedded systems towards implementing multiple functionalities upon a single shared computing platform. This can force tasks of different criticality to share a processor and interfere with each other. We focus on the scheduling of sporadic task systems [3] in these mixed-criticality (MC) systems. The mixed-criticality model that we follow has first been proposed and analyzed, for independent collection of jobs, by Baruah et al. [1]. The model has been extended to task systems by Li and Baruah [5]. The results presented here appear in Baruah et al. [2].

We first describe the model and give some notation. Then, we describe an algorithm (called EDF-VD) to preemptively schedule MC task systems on a single machine. We give a sufficient condition for schedulability by EDF-VD and derive a speed-up guarantee.

The model. Given an integer $K \geq 1$, A K-level MC sporadic task system τ consists of a finite collection $(\tau_1, \ldots, \tau_n)$ of MC sporadic tasks. An MC sporadic task τ_i of a K-level system is characterized by a *criticality level* $\chi_i \in \{1, 2, \ldots, K\}$ and a pair $(c_i, d_i) \in \mathbb{Q}_+^{\chi_i} \times \mathbb{Q}_+$, where: $c_i = (c_i(1), c_i(2), \ldots, c_i(K))$ is a vector of *worst-case execution times* (WCET), we assume that $c_i(1) \leq c_i(2) \leq \ldots \leq c_i(\chi_i)$ and $c_i(\chi_i) = c_i(\chi_i + 1) = \ldots = c_i(K)$; d_i is the *relative deadline* of the jobs of τ_i. We consider *impicit-deadline* tasks in which d_i is equal to the *minimum interarrival time* between two jobs of task τ_i. The *utilization* of task τ_i at level k is defined as $u_i(k) := \frac{c_i(k)}{d_i}$, $i = 1, \ldots, n$, $k = 1, \ldots, K$. The total utilization at level k of tasks that are of criticality level l is $U_l(k) := \sum_{1 \leq i \leq n, \chi_i = l} u_i(k)$, $l = 1, \ldots, K$, $k = 1, \ldots, l$. Task τ_i generates a sequence of jobs $(J_{i1}, J_{i2}, \ldots)$. An MC job J_{ij} of task τ_i is characterized by two parameters: $J_{ij} = (a_{ij}, \gamma_{ij})$, where: $a_{ij} \in \mathbb{R}_+$ is the *arrival time* of the job; $\gamma_{ij} \in (0, c_i(\chi_i)]$ is the *execution requirement* of the job; the *(absolute) deadline* of job J_{ij} is $d_{ij} := a_{ij} + d_i$. It is important to notice that neither the arrival times nor the execution requirements are known in advance. In particular, the value γ_{ij} is discovered by executing the job until it *signals* that it has completed execution. A collection of arrival times and execution requirements is called a *scenario* for the task system. The criticality level of a scenario is defined as the smallest integer $\ell \leq K$ such that $\gamma_{ij} \leq c_i(\ell)$, for each job J_{ij} of each task τ_i. An

*`baruah@cs.unc.edu`. University of North Carolina, USA.
†`vincenzo.bonifaci@iasi.cnr.it`. IASI – Consiglio Nazionale delle Ricerche, Italy.
‡`gianlorenzo.dangelo@gssi.infn.it` Gran Sasso Science Institute (GSSI), Italy.
§`lihaohan@cs.unc.edu` Google, USA.
¶`alberto@dis.uniroma1.it` Sapienza Università di Roma, Italy.
‖`suzanne.vander.ster@vu.nl, leen.stougie@cwi.nl` Vrije U. Amsterdam, the Netherlands.

(online) algorithm correctly schedules a sporadic task system τ if it is able to schedule *every* job sequence generated by τ such that, if the criticality level of the corresponding scenario is ℓ, then all jobs of level at least ℓ are completed between their release time and deadline.

Algorithm EDF-VD. We consider a variant of the Earliest Deadline First algorithm, EDF-VD (EDF with virtual deadlines). Algorithm EDF-VD consists of an *offline preprocessing* phase and a *run-time scheduling* phase. The first phase is performed prior to run time and executes a schedulability test to determine whether τ can be successfully scheduled by EDF-VD or not. If τ is deemed schedulable, this phase also provides two output values that will serve as input for the run-time scheduling algorithm: an integer parameter k (with $1 \leq k \leq K$); and, for each task τ_i of τ, a parameter $\hat{d}_i \leq d_i$, called *virtual deadline*. The second phase performs the actual run-time scheduling and consists of K variants, called EDF-VD(1), ..., EDF-VD(K). Each of these is related to a different value of the parameter k that was provided by the first phase; that is, at run time, the variant EDF-VD(k) is applied. If the scenario is exhibiting a level smaller than or equal to k, then jobs are scheduled according to EDF with respect to the virtual deadlines $(\hat{d}_i)_{i=1}^n$. As soon as the scenario exhibits a level greater than k, jobs are scheduled according to EDF with respect to the original deadlines $(d_i)_{i=1}^n$. The preprocessing phase is based on the following sufficient condition for schedulability by EDF-VD.

Theorem 1 *Given an implicit-deadline task system τ, if either $\sum_{l=1}^{K} U_l(l) \leq 1$ or, for some k $(1 \leq k < K)$, the following condition holds:*

$$1 - \sum_{l=1}^{k} U_l(l) > 0 \quad \text{and} \quad \frac{\sum_{l=k+1}^{K} U_l(k)}{1 - \sum_{l=1}^{k} U_l(l)} \leq \frac{1 - \sum_{l=k+1}^{K} U_l(l)}{\sum_{l=1}^{k} U_l(l)}, \tag{1}$$

then τ can be correctly scheduled by EDF-VD.

Speedup guarantee. The *speedup factor* of a scheduling algorithm A is the smallest real number f such that any task system τ that is feasible on a unit-speed processor is correctly scheduled by A on a speed-f processor. In the following we determine the minimum speedup factor f_K such that any K-level task system that is feasible on an unit-speed processor is correctly scheduled by EDF-VD on a f_K-speed processor. Such problem can be formulated as follows: Find the *largest* q $(q \leq 1)$ such that the following implication holds for all $U_l(k)$, $k = 1, 2, \ldots, K$, $l = k, k+1, \ldots, K$:

$$\sum_{l=k}^{K} U_l(k) \leq q \qquad \forall k = 1, 2, \ldots, K \quad \Rightarrow$$

$$\text{either} \quad \sum_{l=1}^{K} U_l(l) \leq 1 \quad \text{or} \quad \exists k \in \{1, 2, \ldots, K-1\} \text{ s.t.} \begin{cases} 1 - \sum_{l=1}^{k} U_l(l) > 0 \quad \text{and} \\ \dfrac{\sum_{l=k+1}^{K} U_l(k)}{1 - \sum_{l=1}^{k} U_l(l)} \leq \dfrac{1 - \sum_{l=k+1}^{K} U_l(l)}{\sum_{l=1}^{k} U_l(l)}. \end{cases}$$

Number of levels K	Speedup factor f_K
2	1.3333
3	2.0000
4	2.6180
5	3.0811
6	3.7321
7	4.2361

Number of levels K	Speedup factor f_K
8	4.7913
9	5.3723
10	5.8551
11	6.4641
12	6.9487
13	7.5311

Table 1: Minimum speedup factor for $K \leq 13$ levels

If the largest such value of q is q^*, the speedup factor is then $f_K = 1/q^*$. Equivalently, we want to find the *smallest* q such that the above implication does not hold, that is, the premise is true but the conclusion is false; in other words, the largest value of the speedup for which one can still construct a counterexample. This leads to a non-linear formulation that involves disjunctions, which are typically disallowed by numerical solvers. We prove that solving such formulation is equivalent to finding $q^* := \min_{j=1,2,\ldots,K-1} q_j^*$, where each q_j^* is the solution to the non-linear program whose constraints are multivariate polynomial inequalities in the variables $U_l(k)$ and q_j. As such, it can be solved by a (numerical) global non-linear continuous optimization solver. In this case we used Couenne [4]. Couenne was able to find the optimum for any $K \leq 13$. The resulting speedup factors are reported in Table 1.

Theorem 2 *Let τ be a K-level task system with $2 \leq K \leq 13$. If τ is feasible on a unit-speed processor, then it is correctly scheduled by* EDF-VD *on a processor of speed f_K, where f_K ($\pm 10^{-4}$) is as in Table 1.*

References

[1] S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie (2012). *Scheduling real-time mixed-criticality jobs.* IEEE Transactions on Computers, 61(8):1140–1152.

[2] S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie. *Preemptive Uniprocessor Scheduling of Mixed-Criticality Sporadic Task Systems.* Journal of the ACM. To appear.

[3] S. K. Baruah and J. Goossens (2003). *Scheduling real-time tasks: Algorithms and complexity.* In J. Y.-T. Leung, editor, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter 28. CRC Press.

[4] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter (2009). *Branching and bounds tightening techniques for non-convex MINLP.* Optimization Methods & Software 24, 4–5, pages 597–634.

[5] H. Li and S. K. Baruah (2010). *An algorithm for scheduling certifiable mixed-criticality sporadic task systems.* In Proc. 16th IEEE Real-Time Systems Symposium, pages 183–192.