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 $\tau$ consists of a finite collection $(\tau_1, \ldots, \tau_n)$ of MC sporadic tasks. An MC sporadic task $\tau_i$ of a $K$-level system is characterized by a criticality level $\chi_i \in \{1, 2, \ldots, K\}$ and a pair $(c_i, d_i) \in Q^{\chi_i}_+ \times Q_+$, 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 $\tau_i$. We consider implicit-deadline tasks in which $d_i$ is equal to the minimum interarrival time between two jobs of task $\tau_i$. The utilization of task $\tau_i$ 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 $\tau_i$ generates a sequence of jobs $(J_{i1}, J_{i2}, \ldots)$. An MC job $J_{ij}$ of task $\tau_i$ 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 $\gamma_{ij}$ 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 $\tau_i$. An
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 $\tau$ can be successfully scheduled by EDF-VD or not. If $\tau$ 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 $\tau_i$ of $\tau$, 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), \ldots, 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 $\tau$, if either $\sum_{l=1}^K U_l(k) \leq 1$ or, for some $k$ ($1 \leq k < K$), the following condition holds:

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

then $\tau$ 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 $\tau$ 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 $U_l(k)$, $k = 1, 2, \ldots, K$, $l = k, k+1, \ldots, K$: 

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

either $\sum_{l=1}^K U_l(l) \leq 1$ or $\exists k \in \{1, 2, \ldots, K-1\}$ s.t.

$$\left\{ \begin{array}{l}
1 - \sum_{l=1}^k U_l(l) > 0 \quad \text{and} \\
\sum_{l=k+1}^K U_l(k) \leq \frac{1 - \sum_{l=k+1}^K U_l(l)}{\sum_{l=1}^k U_l(l)}.
\end{array} \right.$$
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 $q^* := \min_{j=1,2,...,K-1} q_j^*$, 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 $\tau$ be a $K$-level task system with $2 \leq K \leq 13$. If $\tau$ is feasible on a unit-speed processor, then it is correctly scheduled by EDF-VD on a processor of speed $f_K$, where $f_K (\pm10^{-4})$ is as in Table 1.

References


