Preemptive Uniprocessor EDF Schedulability Analysis with Preemption Costs Considered

Calvin Deutschbein, Sanjoy Baruah
University of North Carolina
{cd, baruah}@cs.unc.edu

Summary: We report here on our ongoing explorations of schedulability and sustainability analysis that is *cognizant of preemption costs*, in the uniprocessor EDF scheduling of sporadic task systems.

I. Introduction

In real-time computing, time bounding processor usage of tasks is critical to ensure that in a running system the deadlines of all tasks are always met. Preemption delays complicate predictability as they are highly sensitive to a variety of system parameters and must be taken into consideration to ensure schedulability of a system, but many existing models do not account for these delays. Several such periodic scheduling problems were studied in [6], and many interesting and important complications and considerations identified and studied; this paper explores some of these complications and considerations as they apply to *sporadic* (see, e.g., [8], [5]) rather than periodic task models. Each *task* τ_i is characterized by a worst case execution time (WCET) C_i, a relative deadline D_i, and a minimum inter-arrival separation T_i. A task is assumed to generate an unbounded sequence of *jobs*, with each job arriving at least T_i apart from any other arrival, with a WCET C_i, and needing to complete execution by a deadline that occurs D_i time-units after arrival. A sporadic *task system* τ comprises a collection of several such independent tasks executing upon a shared platform: $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. A sporadic task system τ in which each task satisfies the additional property that its relative deadline parameter is equal to its period ($D_i = T_i$ for all $\tau_i \in \tau$) is said to be an *implicit-deadline* sporadic task system.

A. Schedulability analysis

Let A denote a scheduling algorithm. A sporadic task system τ is said to be *A-schedulable* on platform P if A schedules all possible release patterns of jobs of τ without missing a deadline on P. An *A-schedulability test* accepts as input the specifications of a sporadic task system and a platform, and determines whether the task system is A-schedulable. An A-schedulability test is said to be *exact* if it identifies all A-schedulable systems, and *sufficient* if it identifies only some A-schedulable systems.

Prior work on EDF-schedulability tests for systems of sporadic tasks on preemptive uniprocessors (see, e.g. [7], [2]) has established that a necessary and sufficient condition for sporadic task system τ to be EDF-schedulable upon a unit-speed processor is the following [2]:

$$\forall t : t \geq 0 : \sum_{\tau_\ell \in \tau} \max\left(0, \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) \times C_i\right) \leq t$$

B. Sustainability

The concept of sustainability [4, page 159] formalizes the expectation that a scheduling algorithm or schedulability test should be more effective upon systems that have at least one task with reduced demand due to excess pessimism in task-wise analysis.

Definition 1 (Sustainable schedulability test [1])**.** *Consider scheduling algorithm A, sporadic A-schedulability test F, A-schedulable by F task system τ, and τ generated job release J. F is said to be a* sustainable *schedulability test for A if and only if algorithm A meets all deadlines when scheduling J even under any of the following changes to jobs in J: (i) decreased execution requirements; (ii) larger relative deadlines; and (iii) later arrival times with the restriction that successive jobs of any task $\tau_i \in \tau$ arrive at least T_i time units apart.*

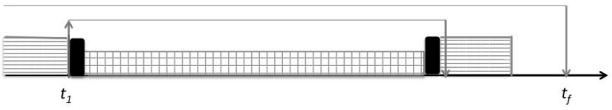
Sustainability of schedulability tests, as defined in Definition 1, provides a guarantee that if a task system is deemed schedulable by the test, it will not fail to meet all deadlines if the system behaves "better" at run time than the specifications. An additional notion of sustainability for schedulability tests, called *self-sustainability*, was defined in [1]:

Definition 2 (self-sustainability [1])**.** *A schedulability test is* self-sustainable *if all task systems with "better" (less constraining) parameters than a task system deemed schedulable by the test are also deemed schedulable by the test.*

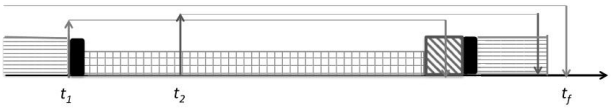
If a self-sustainable schedulability test is used, then sustainable changes will not render schedulable sub-systems unschedulable (or unverifiable).

II. Preemption-cognizant analysis

Figure 1 (a) depicts (with horizontal lines) a job with deadline at t_f that is executing initially. At time-instant t_1, another job (in checked lines) with an earlier deadline arrives and preempts it. The durations devoted to context-switch operations are depicted in the schedule as solid black boxes; there is one such duration when the job is preempted, and another when the preempted job is resumed. Let us refer to these costs as the "context-switch *save*" and "context-switch *restore*" costs respectively. With regards to the sporadic task model



(a) At time-instant t_1, an executing job with deadline at t_f is preempted by a newly-arrived job with an earlier deadline.



(b) A job with deadline between those of the preempting and preempted jobs arrives at time-instant t_2.

Fig. 1. Time-line diagrams illustrating preemption of the job with deadline at time-instant t_f.

discussed above, let us enhance the model by associating with each sporadic task τ_i the additional parameters $\mathrm{cs}_i, \mathrm{css}_i$, and csr_i, satisfying the relationship $(\mathrm{cs}_i = \mathrm{css}_i + \mathrm{csr}_i)$. Here, css_i denotes the maximum duration of time needed to save the context associated with an executing job of τ_i that is preempted, while csr_i denotes the maximum duration of time needed to restore this context when the job is resumed; cs_i denotes the maximum total overhead associated with each preemption of a job of τ_i.

For the remainder of this paper assume as a notational convenience, with no loss of generality, that tasks are indexed by non-decreasing D_i parameters.

Observation 1. *Any preemption operation requires that two sub-operations be performed: (i) a context-switch save operation when preemption occurs, and (ii) a context-switch restore operation when the preempted job later resumes.*

1) *The save operation occurs before the preempting job's execution and preempting job's deadline.*
2) *The restore operation need not complete prior to the deadline of the preempting job.*
3) *The preempting job may not always be the one that restores the context.*

The third observation above is illustrated in Figure 1(b).

III. Sustainability results

We now examine sustainability properties of preemptive uniprocessor EDF in scheduling sporadic task systems with respect to periods, WCET's, context switch costs, and relative deadline parameters of the individual tasks.

Theorem 1. *The EDF scheduling of sporadic task systems is sustainable with respect to WCET parameters and context switch costs.*

Proof. Given a task system τ with context switch costs, we define a transformation from τ to a new task system τ' in which all context switch costs are equal to zero:

- For any instance of τ, consider each job released by each task, and each context switch operation (save/store separately) to be unique jobs.
- For jobs of τ, create a job with the same release, deadline, and WCET.
- For saves, create a job with WCET and relative deadline equal to the context-switch save cost, and release equal to preempting job's release (hence the job is non-preemptable in any feasible schedule).
- For each restore operation, create a job with WCET equal to the context-switch restore cost, release equal to the release of the preempting job, and deadline earlier than the preempted job's deadline by some arbitrarily small $\epsilon > 0$

Consider EDF on τ'. As a simple application of uniprocessor EDF, since τ is schedulable and EDF is optimal (see, e.g., [3, Sec 4.1 (p 29)])., EDF schedules τ' as well.

A reduction in the WCET of any job in τ' may model a reduction in task WCET, task context-switch save cost, or task context-switch restore cost. EDF is sustainable in the no context switch cost case, so τ' is still schedulable.

The schedule produced by τ' is valid on a similarly modified τ, so EDF is sustainable with respect to task WCET and context switch costs. □

Theorem 2. *Implicit-deadline sporadic task systems are <u>not</u> sustainable with respect to period parameters even if all* cs_i *values are equal.*

Proof. Consider the following task system:

τ_i	C_i	T_i	cs_i
τ_1	5	10	3
τ_2	5	10	3

Both tasks have the same period parameter, so no premptions occur and the system is schedulable under EDF.

If however τ_2's period is increased to 12, τ_2's job arrives at time zero and τ_1's at time-instant 1, then τ_2's job gets preempted; this results in a deadline miss. □

IV. A sufficient EDF schedulability test

We start out with a couple of lemmas characterizing preemptions. The following result is well known in the folk-lore of real-time scheduling theory (see, e.g., [3, page 22]):

Lemma 1. *The total number of preemptions in any EDF schedule is no greater than the number of jobs that were executed (minus one).*

The following lemma identifies a further relationship between any preempted job, and the job that preempts it:

Lemma 2. *No job of τ_i is preempted by any job of τ_j, for any $j > i$.*

Proof. Observe that in order for a job j_1 to be preempted by another job j_2 at some time-instant t_o, it is necessary that

1) Job j_1 be executing at time-instant t_o;
2) Job j_2 arrive at time-instant t_o; and
3) the deadline of j_2 is earlier than the deadline of j_1.

The third condition above may only be satisfied if the task that generated job j_2 has a smaller relative deadline than the task that generated job j_1; the lemma follows from the condition that the tasks are indexed in non-decreasing order of relative deadline parameters. □

As was pointed out in Observation 1 (Section II), there are two operations associated with each preemption. The context-switch save operation must occur before the deadline of the preempting job, but the context-switch restore operation may occur after this deadline. Based upon Observation 1, it should be clear that it is safe to "charge" the cost of the context-switch save operation to the preempting job (in the sense of inflating the execution time of the preempting job by the amount of time required for the context-switch save operation, and then assuming that the context-switch store operation takes no time). But what about the context-switch restore operation? — as is evident from Figure 1 (b), this operation may take place well after the deadline of the preempting job. We will argue in Lemma 3 below that it is safe to also charge this cost to the preempting job. That is,

- Suppose that an executing job j is preempted by a job j', and let $\mathrm{css}(j)$ and $\mathrm{csr}(j)$ denote the durations of time required to save the context upon preemption, and to restore the context at a later point in time respectively.
- We will inflate the execution requirement of j' by an amount $(\mathrm{css}(j) + \mathrm{csr}(j))$, and then perform EDF schedulability-analysis of the resulting collection of jobs under the assumption that preemptions incur no overhead.
- If this EDF schedulability-analysis does not detect any deadline misses, then there would have been no deadline misses in the original schedule – the one in which preemptions do incur costs.

This is formally stated in the following lemma

Lemma 3. *If a sporadic task system is not EDF-schedulable when preemption durations are considered, then it remains unschedulable by EDF if* (i) *the execution time of each preempting job is inflated by the sum of the context-switch save and context-switch restore durations of the preempted jobs; and* (ii) *the preemption durations are subsequently assumed to be equal to zero.*

Proof. Suppose that sporadic task system is unschedulable by EDF when preemption durations are considered. Let I denote some minimal collection of jobs generated by τ, upon which EDF misses a deadline — by *minimal*, we mean that EDF would not miss any deadlines upon scheduling any proper subset of the jobs in I. Without loss of generality, assume that the earliest release time of any job in I is time zero, and let t_f denote the time-instant at which the deadline miss occurs.

- Since a deadline is missed at t_f, it must be the case that the processor is completely busy over the interval $[0, t_f)$.
- The transformation defined in the lemma effectively moves the duration of each context-switch restore to *before* the deadline of the preempting job that caused it — hence, the amount of execution needed by any time-instant can only *increase* upon applying the transformation.
- When the transformation is applied to the jobs in I, therefore, it must remain the case that the processor will be completely busy over the interval $[0, t_f)$, and the deadline that was missed in the original schedule is missed in the schedule of the transformed collection of jobs as well.

The lemma follows. □

Lemmas 1-3 above allow us to define a "safe" transformation from the problem of EDF-schedulability analysis when preemption durations are non-zero to the problem of EDF-schedulability analysis when preemption durations are all equal to zero:

1) By Lemma 1, each job is responsible for at most one preemption.
2) By Lemma 2, a job of τ_i may only be responsible for a preemption of a job of τ_j for some $j > i$.
3) By Lemma 3, therefore, it is safe to inflate the execution requirement of each job of τ_i by the maximum of the preemption durations associated with any task τ_j, $j > i$ (i.e., a quantity $\max_{j>i}\{\mathrm{cs}_j\}$), and subsequently consider all preemption costs to be equal to zero.

This argument is formalized in the following theorem.

Theorem 3. *A sporadic task system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, in which each task $\tau_i = (C_i, D_i, T_i, \mathrm{cs}_i)$ is characterized by its WCET, relative deadline, period, and context-switch costs, is EDF-schedulable if the sporadic task system $\tau' = \{\tau'_1, \tau'_2, \ldots, \tau'_n\}$ is EDF-schedulable, with each $\tau'_i = (C'_i, D_i, T_i, 0)$ and the C'_i's defined as follows:*

$$C'_i \leftarrow C_i + \max_{j>i}\{\mathrm{cs}_j\} \tag{1}$$

V. Ongoing Work

We are currently working on developing less conservative sufficient algorithms for preemption-cognizant schedulability analysis of sporadic task systems. We are also exploring computational complexity issues – although it is immediately evident (since it follows from prior results concerning preemption-oblivious schedulability analysis) that preemption-cognizant analysis is NP-hard for sporadic task systems in general, we do not yet know whether such analysis is intractable or not for *implicit-deadline* sporadic task systems.

REFERENCES

[1] Theodore Baker and Sanjoy Baruah. Sustainable multiprocessor scheduling of sporadic task systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Dublin, July 2008. IEEE Computer Society Press.

[2] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, Orlando, Florida, 1990. IEEE Computer Society Press.

[3] Sanjoy Baruah, Marko Bertogna, and Giorgio Buttazzo. *Multiprocessor Scheduling for Real-Time Systems*. Springer Publishing Company, Incorporated, 2015.

[4] Sanjoy Baruah and Alan Burns. Sustainable scheduling analysis. In *Proceedings of the IEEE Real-time Systems Symposium*, pages 159–168, Rio de Janeiro, December 2006. IEEE Computer Society Press.

[5] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Second edition, 2005.

[6] J. Goossens T. Chapeaux G. Phavorin, P. Richard and C. Maiza. Scheduling with preemption delays: anomalies and issues. In *Proceedings of the 23rd International Conference on Real-Time and Network Systems (RTNS)*, 2015.

[7] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[8] Aloysius Mok. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.