# Adapting a Fixed Priority Assignment Algorithm to Real-time Embedded Systems with Cache Memory

Hai-Nam Tran, Stéphane Rubini, Frank Singhoff, and Jalil Boukhobza
Univ. Bretagne Occidentale, UMR 6285, Lab-STICC, F-29200 Brest, France
Email: {hai-nam.tran,rubini,singhoff,boukhobza}@univ-brest.fr

## Abstract

*Handling cache related preemption delay in preemptive scheduling context for real-time embedded systems still stays an open issue despite of its practical importance. Classical priority assignment algorithms and feasibility tests are usually based on the assumption that preemption cost is negligible. Then, a system that could be schedulable at design time can fail to meet its timing constraints in practice due to preemption cost. In this article, we propose an approach to adapt a fixed priority assignment algorithm to real-time embedded systems with cache memory. Our algorithm addresses the effect of cache related preemption delay while assigning priorities to tasks and thus guarantees the schedulability and feasibility of a system.*

## 1 Introduction

Integrating cache memory in real-time embedded systems (RTES) generally improves the overall system performance, but unfortunately it can lead to execution time variability due to the variation of preemption cost. When a task is preempted by a higher priority task, memory blocks belonging to the task could be removed from the cache. Once this task resumes, previously removed memory blocks have to be reloaded. Thus, a new preemption cost named *Cache Related Preemption Delay* (CRPD) is introduced. By definition, CRPD is the additional time to refill the cache with memory blocks evicted by preemption. In [1], the authors showed that CRPD can present up to 40% of the Worst Case Execution Time of a program.

*Problem statement:* CRPD introduces two issues. First, it accounts a high proportion of the preemption cost and causes the preemption cost to become not negligible [1]. Second, in the context of preemptive fixed priority scheduling, there is no priority assignment algorithm which takes CRPD into account and can guarantee the schedulability of a system. As far as we know, existing priority assignment algorithms only focused on the system model in which preemption cost is assumed to be zero.

*Contributions:* We propose an approach to perform priority assignment and to guarantee that a system is schedulable while taking into account CRPD. To achieve this, we extend the priority assignment and feasibility test in [2]. Our approach consists in computing the interference from computational requirements and CRPD of higher priority tasks when assigning a priority level to a task.

## 2 Approach

In this section, we present our approach and discuss the raised issues. First, we present the priority assignment algorithm and explain the need of an appropriate feasibility test. Second, we present how a feasibility test is extended in order to take into account CRPD.

We assume a uniprocessor system with one level of direct-mapped instruction cache. There are $n$ tasks scheduled by a fixed priority preemptive scheduler.

### 2.1 Priority Assignment

We choose to extend Audsley's Optimal Priority Assignment (OPA) algorithm [2]. This algorithm allows us to assign a certain priority level to a task and to immediately verify its feasibility, which is the phase where CRPD is taken into account.

We have $n$ priority levels corresponding to $n$ tasks, with $n$ being the lowest priority level. The algorithm starts by assigning the lowest priority level to a given task $\tau_i$.

If $\tau_i$ is not schedulable at priority level $n$, the algorithm assigns the priority level $n$ to a different task. If $\tau_i$ is schedulable, the algorithm assigns priority level $n$ to $\tau_i$ and moves to the next priority level $n-1$. Then, it checks whether a task in the set of unassigned priority tasks is feasible with the $n-1$ priority level. The algorithm continues until all tasks are assigned a priority level. If there are not any schedulable tasks at a given priority level, the system is not schedulable and the algorithm terminates.

In the original work of Audsley [2], the feasibility test was designed with two system properties. First, preemption cost is assumed to be zero. Second, the response time of a task is not affected by the priority ordering of higher priority tasks The two properties are not true anymore when CRPD is taken into account. As a result, we need to design an appropriate feasibility test. This test must be able to verify the feasibility of a task under a given priority level while the complete priority assignment of higher priority tasks is not achieved. To the best of our knowledge, there is no existing work dealing with such an issue.

### 2.2 Feasibility Test

Regarding the feasibility test in [2], a task is schedulable if all its jobs released during the feasibility interval can meet their deadlines. We recall that a feasibility interval is an interval for which testing of task feasibility is needed. Assuming a given job of task $\tau_i$ released at time $t$, which requires $C_i$ units of computation time and must complete before $D_i$, this job experiences interference $I_i^t$ due to by other jobs of higher priority tasks during the interval $[t, t + D_i)$. The job of the task $\tau_i$ is feasible if:

$$C_i + I_i^t \leq D_i \tag{1}$$

A task $\tau_i$ is feasible if for all jobs of $\tau_i$ released at time $t$ in the feasibility interval, equation 1 is satisfied.

$I_i^t$ is computed by taking into account the interference from each job of higher priority tasks. The interference from a job of higher priority task $\tau_j$ is made up of its capacity $C_j$. In systems with cache, we also have to take into account the CRPD created by this job.

Interference from a job of task $\tau_j$ consists of three parts. The first part is the capacity of task $\tau_j$. The second part is the preemption cost due to task $\tau_j$ preempting task $\tau_i$. The third part is the preemption cost due to task $\tau_j$ preempting intermediate priority tasks $\tau_k$, $\tau_k$ has a higher priority than $\tau_i$ but a lower priority than $\tau_j$, i.e. preemption cost due to nested preemption.

We denote the second and third parts as CRPD Interference. In order to compute CRPD interference, one needs to evaluate: (1) the number of preemptions and (2) the CRPD of each preemption.

**Number of Preemptions:** The computation of the number of preemptions is the first issue. In OPA, when verifying the feasibility of a task at a given priority level, we only assumed that other tasks have higher priority without a complete priority assignment. As a result, the occurrence of a preemption between jobs of those tasks cannot be determined. Thus, exact computation of the number of preemptions in the interval $[t - T_i + D_i, t + D_i)$ cannot be achieved.

**Preemption cost - CRPD:** The computation of the CRPD of each preemption is the second issue. In our work, a cache access profile of a task is modeled by a set of useful cache blocks (UCB) and a set of evicting cache blocks (ECB). Detailed explanation of the CRPD computation and execution model could be found in [3].

Assuming that the cache access profiles of tasks are preliminary computed, we can compute the CRPD of a preemption if preempting task and preempted tasks are identified. However, if task priorities are not completely assigned, we cannot identify intermediate priority tasks for a given preempting tasks.

We propose three solutions to compute an upper-bound of the number of preemptions and the CRPD of each preemption. They have different degrees of pessimism, different complexities, and give different results in terms of schedulable task sets coverage.

The first solution is adding the worst-case effect of CRPD to the capacity of higher priority tasks. In this solution, all jobs of higher priority tasks are considered as preemptions with the worst-case effects. The second solution consists of evaluating all possible potential preemptions and computing an upper-bound CRPD of each preemption. By definition, a potential preemption amongst jobs of higher priority tasks without a complete priority assignment is a preemption that may occur when a job is released while other jobs did not complete their executions. The third solution is evaluating all possible preemption sequences caused by jobs of higher priority tasks.

Exhaustive experimentations by scheduling simulations with randomly generated task sets were achieved to evaluate our approach in terms of complexity and efficiency. The result shows that if a task set is found schedulable by our approach, it is practically schedulable when CRPD is taken into account. In our experiments, at the processor utilization of 75%, 100% task sets are concluded to be schedulable by OPA while only 82% of them are schedulable when CRPD is taken into accouunt. Our algorithms successfully eliminates non-schedulable task sets. Detailed explanation and experiment results could be found in [3]. Our work is implemented in Cheddar - an open-source scheduling analyzer [4].

## 3 Conclusions

In this article, we presented an approach to perform priority assignment with CRPD taken into account. Our approach consists in extending the priority assignment algorithm and the original feasibility test in [2]. We presented two issues regarding the computation of the number of preemption and the preemption cost. In our work, three solutions were proposed and implemented into Cheddar to solve these issues. Exhaustive experimentations by scheduling simulation have shown that our approach provides a mean to guarantee the schedulability of real-time embedded systems with cache memory.

## References

[1] R. Pellizzoni and M. Caccamo, "Toward the predictable integration of real-time cots based systems," in *28th International Real-Time Systems Symposium (RTSS)*. IEEE, 2007, pp. 73–82.

[2] N. C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times," in *Technical Report YCS 164, Dept. Computer Science*. University of York, UK, 1991.

[3] H.-N. Tran, F. Singhoff, S. Rubini, and J. Boukhobza, "Addressing cache related preemption delay in fixed priority assignment," in *20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2015, pp. 1–8.

[4] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *ACM SIGAda Ada Letters*, vol. 24, no. 4, 2004, pp. 1–8.