Optimal FP Scheduling with Deferred Pre-emption

Robert I. Davis (Speaker) * Marko Bertogna †

1 Introduction

A common misconception about fixed priority scheduling of sporadic tasks on a single processor is that fully pre-emptive scheduling is the best approach in terms of schedulability. Fixed priority non-pre-emptive scheduling (FPNS) and fixed priority pre-emptive scheduling (FPPS) are however incomparable; there are tasksets that are schedulable under FPNS that are not schedulable under FPPS and vice-versa.

The term *fixed priority scheduling with deferred pre-emption* (FPDS) has been used to refer to a variety of techniques by which pre-emptions may be deferred for some period of time after a higher priority task becomes ready [4]. In this paper, we assume a form of FPDS where each task has a final non-pre-emptive region (FNR). If this region is of the minimum possible length[1] for all tasks, then we have FPPS, whereas if the FNR constitutes all of the task's execution time then we have FPNS; thus FPDS is a superset of, and dominates both FPPS and FPNS.

With FPDS, there are two key parameters that affect taskset schedulability: the priority assigned to each task, and the length of each task's FNR. The length of the FNR affects both the schedulability of the task itself, and the schedulability of tasks with higher priorities. This is a trade-off as increasing the length of the FNR can improve schedulability for the task itself by reducing the number of times it can be pre-empted, but potentially increases the blocking experienced by higher priority tasks reducing their schedulability. Here, we present an optimal algorithm for FPDS. This *Final Non-pre-emptive Region and Priority Assignment* (FNR-PA) algorithm is optimal in the sense that it is guaranteed to find a combination of priority assignment and FNR lengths that result in a schedulable system under FPDS whenever such a schedulable combination of these parameters exists. Full details are given in [4].

2 System Model and Analysis

We consider the fixed priority scheduling of a set of sporadic tasks (or *taskset*) on a single processor. Each taskset comprises a static set of n tasks ($\tau_1 \ldots \tau_n$). We assume that the index i of task τ_i represents the task priority, hence τ_1 has the highest priority, and τ_n

*rob.davis@york.ac.uk. Real-Time Systems Research Group, Department of Computer Science, University of York, York, UK.

†marko.bertogna@unimore.it. Algorithmic Research Group, Department of Mathematics, University of Modena, Italy.

[1]The minimum possible length of a non-pre-emptive region is 1 rather than 0, as we assume a discrete time model and tasks cannot be pre-empted during a processor clock cycle.

the lowest. We assume a discrete time model, where all task parameters are assumed to be positive integers. Each task τ_i is characterized by its worst-case execution time C_i, minimum inter-arrival time or period T_i, and relative deadline D_i. Each task τ_i gives rise to a potentially unbounded sequence of jobs, each of which has an execution time upper bounded by C_i, an arrival time at least T_i after the arrival of its previous job, and an absolute deadline that is D_i after its arrival. Under FPDS, each task is assumed to have a final non-pre-emptive region of length F_i in the range $[1, C_i]$. The worst-case response time R_i of a task is given by the longest possible time from release of the task until it completes execution. Thus task τ_i is schedulable if and only if $R_i \leq D_i$, and a taskset is schedulable if and only if $\forall i \; R_i \leq D_i$.

We now recapitulate schedulability analysis for FPDS for sporadic tasksets given by Bril et al. [3]. They showed that for FPDS, the longest response time of a task τ_i occurs for some job of that task within the priority level-i active period starting at a δ-critical instant. Lemma 3 in [3] states that the worst-case length of a priority level-i active period A_i is given by the minimum solution to the following fixed point equation:

$$A_i \; = \; B_i^{FNR} \; + \; \sum_{j \in \mathbf{hep}(i)} \left\lceil \frac{A_i}{T_j} \right\rceil C_j. \tag{1}$$

In (1) the term B_i^{FNR} is the longest time that task τ_i can be blocked from executing by lower priority tasks, and is given by $B_i^{FNR} \; = \; \max_{l \in \mathbf{lp}(i)} (F_l - 1)$. The number of jobs G_i of task τ_i in the priority level-i active period is given by $G_i \; = \; \lceil A_i / T_j \rceil$. The start time $W_{i,g}$ of the final non-pre-emptive region of job g (where $g = 0$ is the first job) measured with respect to the start of the δ-critical instant is given by the minimum solution to the following fixed point equation:

$$w_{i,g}^{m+1} \; = \; B_i^{FNR} \; + \; (g+1)C_i \; - \; F_i \; + \; \sum_{j \in \mathbf{hp}(i)} \left(\left\lfloor \frac{w_{i,g}^m}{T_j} \right\rfloor + 1 \right) C_j. \tag{2}$$

To find the worst-case response time, the start times of the final non-pre-emptive regions $W_{i,g}$ need to be calculated for jobs $g = 0, 1, 2, 3, \ldots G_i - 1$. The worst-case response time of task τ_i is then given by: $R_i \; = \; \max_{g=0,1,2,3,\ldots G_i - 1} W_{i,g} + F_i - gT_i$. Task τ_i is schedulable provided that $R_i \leq D_i$.

3 Optimal FPDS

Definition 1 (Optimality) *An algorithm Z is said to be optimal for FPDS if there are no tasksets compliant with the task model that are schedulable under FPDS with some priority assignment and some set of values for the lengths of the final non-pre-emptive regions of each task, that are not also schedulable using the priority assignment and set of lengths for the final non-pre-emptive regions determined by algorithm Z.*

Theorem 2 *The Final Non-pre-emptive Region Priority Assignment (FNR-PA) algorithm (Algorithm 1) is optimal for the FNR-PA problem.*

Theorem 3 *For any taskset where there exists a priority ordering and a set of final non-pre-emptive region lengths that is schedulable under FPDS, the FNR-PA algorithm results in a blocking factor B_i^{FNR} at every priority level i that is no larger than that obtained with any other schedulable priority and final non-pre-emptive region length assignment.*

```
for each priority level k, lowest first {
  for each unassigned task τ {
     determine the smallest length F(k) for the FNR of task τ such that
     it is schedulable at priority k, assuming all other unassigned tasks
     have higher priorities. Record as task y the unassigned task with
     the minimum FNR length at priority k.
  }
  if no tasks are schedulable at priority k {
     return unschedulable
  } else {
     assign task y priority k using the value of F(k) as its FNR length.
  }
}
return schedulable
```

Figure 1: FNR-PA Algorithm

Proof of Theorems 2 and 3 is given in [4] and will be presented at the conference.

The FNR-PA algorithm (which is based on Audsley's Optimal Priority Assignment algorithm [1]) effectively requires a maximum of $n(n+1)/2$ pseudo-polynomial task schedulability tests to determine an optimal priority and final non-pre-emptive region length assignment, which compares favourably with a search space of size $n! \prod_{\forall i} C_i$. Thus, the FNR-PA algorithm represents a significant reduction in complexity, making the problem tractable for the majority of practical applications.

4 Summary

Fixed priority scheduling with deferred pre-emption (FPDS), dominates both fixed priority fully pre-emptive (FPPS) and fixed priority non-pre-emptive scheduling (FPNS).

The main contribution of this work is the introduction of an optimal algorithm for FPDS. This FNR-PA algorithm is optimal in the sense that it is guaranteed to find a combination of priority assignment and task final non-pre-emptive region lengths that result in a schedulable system under FPDS, whenever such a schedulable combination of these parameters exists. As a consequence of optimising schedulability under FPDS, the FNR-PA algorithm has the notable side-effect that for any given taskset, it minimises the blocking effect due to final non-pre-emptive regions at every priority level. Using an analytical method of computing final non-pre-emptive region lengths, given in [4], the FNR-PA algorithm requires at most $n(n+1)/2$ (pseudo-polynomial) task schedulability tests to find an optimal solution, making the problem tractable for the majority of practical real-time systems.

References

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

[2] G.C. Buttazzo, M. Bertogna, G. Yao (2013) *Limited Preemptive Scheduling for Real-Time Systems: A Survey* IEEE Transactions on Industrial Informatics. In press. Downloadable from http://retis.sssup.it/ marko/publi.html

[3] R. Bril, J. Lukkien, and W. Verhaegh (2009) *Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption* Real-Time Systems, 42(1-3):63119.

[4] R.I. Davis, M. Bertogna (2012) *Optimal Fixed Priority Scheduling with Deferred Pre-emption* In proceedings 33rd IEEE Real-Time Systems Symposium.