Improvements in the configuration of Posix 1003.1b scheduling

Mathieu Grenier, Nicolas Navet

Improvements in the configuration and analysis of Posix 1003.1b scheduling

Mathieu Grenier Nicolas Navet
LORIA-INRIA
Campus Scientifique, BP 239
54506 Vandoeuvre-lès-Nancy- France
{grenier, nnavet}@loria.fr

Abstract

Posix 1003.1b compliant systems provide two well-specified scheduling policies, namely sched_rr (Round-Robin like) and sched_fifo (FPP like). Recently, an optimal priority and policy assignment algorithm for Posix 1003.1b has been proposed in the case where the quantum value is a system-wide constant. Here we extend this analysis to the case where quanta can be chosen on a task-per-task basis. The algorithm is shown to be optimal with regards to the power of the feasibility test (i.e. its ability to distinguish feasible and non feasible configurations). Though much less complex than an exhaustive exploration, the exponential complexity of the algorithm limits its applicability to small or medium-size problems. In this context, as shown in the experiments, our proposal allows achieving a significant gain in feasibility over FPP and Posix with system-wide quanta, and therefore using the computational resources at their fullest potential.

1. Introduction

Context of the paper. This study deals with the scheduling of real-time systems implemented on Posix 1003.1b compliant Operating System (OS). Posix 1003.1b [7], previously known as Posix4, defines real-time extension to Posix mainly concerning signals, inter-process communications, memory mapped files, synchronous and asynchronous IO, timers and scheduling (a recap of Posix's features related to scheduling is given in §2.1). This standard has become very popular and most of today's OS conform, at least partially, to it.

Problem definition. Posix 1003.1b compliant OSs provide two scheduling policies *sched_fifo* and *sched_rr*, which under some restrictions discussed in §2.1, are respectively equivalent to Fixed Preemptive Priority (FPP) and Round-Robin (RR for short). Thus, under Posix 1003.1b, each process is assigned both a priority, a scheduling policy and, in the case of Round-Robin, a quantum. At each point in time, one of the ready processes with the highest priority is executed, according to the rules of its scheduling policy (e.g. yielding the CPU after a quantum under RR).

The problem addressed here is to assign priorities, policies and quanta to tasks in such a way as to respect deadline constraints. For FPP alone, the well-known Audsley algorithm [2] is optimal. A similar algorithm exists for both RR and FPP in the case of a system-wide quantum [6]. Here we consider the case where quanta can be chosen on a task-per-task basis. As it will be seen in §3.2, the complexity of the problem is such that an exhaustive search is usually not feasible even on small size problems. For instance, a task set of cardinality 10 with quanta chosen among 5 different values requires to analyze the feasibility of more 10^{11} different configurations (see §3.2).

Contributions. Traditionally, the RR policy is only considered useful for low priority processes performing some background computation tasks "when nothing more important is running". In this paper, as we did in [10, 6], we argue that the combined use of RR and FPP allows to successfully schedule a large number of systems that are unschedulable with FPP alone.

The contribution of the paper is twofold, first we propose an algorithm for assigning priorities, policies and quanta that is optimal in the sense that if there exists at least a feasible solution[1], then the algorithm will return a feasible solution. The algorithm being an extension of the classical Audsley algorithm [2] and the *Audsley-RR-FPP* from [6], we name it the *Audsley-RR-FPP** algorithm. The worst-case complexity of the algorithm is assessed and a set of optimizations are proposed to reduce the search space. The second contribution of the paper is that we give further evidences that the combined use of both FPP and RR is effective - especially when quanta can be chosen for each individual task - for finding feasible schedules even when the workload of the system is high.

Related work. We identify two closely related lines of research: schedulability analyses and priority assignment.

[1] We call here a *feasible* solution, a solution that successfully passes a schedulability test verifying property 2 (see §2.5). In the following, we make use of the response time bound analysis derived in [9].

Audsley in [2, 3] proposes an optimal priority assignment algorithm for FPP, that is now well-known in the literature as the Audsley algorithm. Later on in [5], this algorithm has been shown to be also optimal for the non-preemptive scheduling with fixed priorities. The problem of best assigning priorities and policies under Posix 1003.1b was first tackled in [9] but the solution relies on heuristics and is not optimal in the general case. Then, in [6], an optimal solution is proposed for the case where the quantum value is a system-wide constant.

As in [6], the problem addressed here is different than in the plain FPP case because the use of RR leads to the occurrence of scheduling "anomalies", which are sometimes counter-intuitive. For instance, as it will be seen in §2.5, increasing the quantum value for a task can leads sometimes to a greater worst-case response time for this task. Similarly, decreasing the set of higher priority tasks, can increase the response time (see [6]). This prevents us from using the proposed priority and assignment algorithm with the schedulability assessed by simulation, or with a feasibility test that would not possess some specific properties discussed in §2.5. Indeed there would be cases where the algorithm would discard schedulable assignments and thus not be optimal. In this study, feasibility is assessed by the analysis published in [9], which ensures that the computed response time bounds decrease when the set of higher priority tasks is reduced. This property enables us to use an Audsley-like algorithm for the assignment that will be shown to be optimal with regard to the power of the test, that is its ability to distinguish feasible or non feasible configurations.

Organisation. Section *2* summarizes the main features of the scheduling under Posix 1003.1b and introduces the model and notations. In section 3, we present the optimal priority, policy and quantum assignment *Audsley-RR-FPP** algorithm. Efficiency of the proposal is then assessed in section 4.

2. Scheduling under Posix 1003.1b: model and basic properties

In this section we present the system model and summarize the main features related to scheduling of Posix 1003.1b. We then present the assumptions made in this study and derive some basic properties of the scheduling under Posix 1003.1b that will be used in the subsequent sections.

2.1. Overview of Posix 1003.1b scheduling

In the context of OS, we define a task as a recurrent activity which is either performed by repetitively launching a process or by a unique process that runs in cycle. Posix 1003.1b specifies 3 scheduling policies: *sched_rr*, *sched_fifo* and *sched_other*. These policies apply on a process-by-process basis: each process run with a particular scheduling policy and a given priority. Each process inherits its scheduling parameters from its father but may also change them at run-time.

- *sched_fifo* : fixed preemptive priority with First-In First-Out ordering among same-priority processes. In the rest of the paper, it will be assumed that all *sched_fifo* tasks of an application have different priorities. With this assumption and without change during run-time *sched_fifo* is equivalent to FPP.
- *sched_rr* : Round-Robin policy (RR) which allows processes of the same priority to share the CPU. Note that a process will not get the CPU until a higher priority ready-to-run processes are executed. The quantum value may be a system-wide constant (e.g. QNX OS), process specific (e.g. VxWorks OS) or fixed for a given priority interval.
- *sched_other* is an implementation-defined scheduler. It could map onto *sched_fifo* or *sched_rr*, or also implement a classical Unix time-sharing policy. The standard merely mandates its presence and its documentation. Because we cannot rely on the same behaviour of *sched_other* under all Posix compliant OSs, it is strongly suggested not to use it if a portability is a matter of concern. We will not consider it in our analysis.

Associated with each policy is a priority range. Depending on the implementation, these priority ranges may or may not overlap but most implementations allow overlapping. Note that these previously explained scheduling mechanisms similarly apply to Posix threads with the system contention scope as standardised by Posix 1003.1c standard [7].

2.2. System model

The activities of the system are modeled by a set $\mathcal{T}$ of n periodic and independent tasks $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_n\}$. Each task τ_i is characterized by a tuple (C_i, T_i, D_i) where each request of τ_i, called an instance, has an execution time of C_i, a relative deadline D_i and a period equal to T_i time units. One denotes by $\tau_{i,j}$ the j^{th} release of τ_i. As usual, the response time of an instance is the time elapsed between its arrival and its end of execution.

Under Posix 1003.1b, see §2.1, each task τ_i possesses both a priority p_i and a scheduling policy $sched_i$. In this study, we choose the convention "the smaller the numerical value, the higher the priority". In addition to the priority, under RR, each task τ_i is assigned a quantum value ψ_i. The priority and scheduling policy assignment $\mathcal{P}$ is fully defined by a set of n tuples $(\tau_i,\ p_i, sched_i^{\mathcal{P}})$ (i.e. one for each task). A quantum assignment under $\mathcal{P}$, denoted by $\Psi_{\mathcal{P}}$, defines the set of quantum values $\psi_i^{\Psi_{\mathcal{P}}}$ where $\psi_i^{\Psi_{\mathcal{P}}}$ is the quantum of τ_i. The whole scheduling is fully defined by the tuple $(\mathcal{P}, \Psi_{\mathcal{P}})$ which is called a *configuration* of the system.

Under assignment $\mathcal{P}$, the set of tasks $\mathcal{T}$ is partitioned into separate layers, one layer for each priority level j

where the layer $\mathcal{T}_j^{\mathcal{P}}$ is the subset of tasks assigned to priority level j. Under $\mathcal{P}$, $\mathcal{T}_{hp(j)}^{\mathcal{P}}$ (resp. $\mathcal{T}_{lp(j)}^{\mathcal{P}}$) denotes the set of all tasks possessing a higher (resp. lower) priority than j. A layer in which all tasks are scheduled with RR (resp. FPP) is called an RR layer (resp. FPP layer). In the following, $\mathcal{P}$ or $\Psi_{\mathcal{P}}$ will be omitted when no confusions are possible. A list of the notations is provided in appendix at the end of the paper.

In the following, a task τ_i is said *schedulable* under assignment $(\mathcal{P}, \Psi_{\mathcal{P}})$ if its response time bound, as computed by the existing Posix 1003.1b schedulability analysis [9], is no greater than its relative deadline (i.e. maximum duration allowed between the arrival of an instance and its end of execution). The whole system is said schedulable if all tasks are schedulable. Note that the test presented in [9] is sufficient but not necessary, there are thus task sets which won't be classified as schedulable while there exist configurations under which no deadlines are missed.

2.3. Assumptions

In this study, as explained in *§2.1*, only *sched_fifo* and *sched_rr* are considered for portability concern. Due to the complexity of assigning priorities and scheduling policies, the following restrictions are made:

1. context switch latencies are neglected, but they could be included in the schedulability analysis of [9] as classically done (see, for instance, [11]),
2. since a priority level without any tasks has no effect on the scheduling, we impose the priority range to be contiguous,
3. two tasks having different scheduling policies have different priorities, i.e., $\forall i \neq j, sched_i \neq sched_j \implies p_i \neq p_j$,
4. all *sched_fifo* tasks must possess distinct priorities ($sched_i = sched_j = sched\_fifo \implies p_i \neq p_j$). With this assumption and without priority change at run-time, *sched_fifo* is equivalent to fixed-preemptive priority (FPP). Thus, several tasks having the same priority are necessarily scheduled under *sched_rr* policy,
5. the quantum value can be chosen on a task-per-task basis in the interval $[\Psi_{\min}, \Psi_{\max}]$, where $\Psi_{\min}$ and $\Psi_{\max}$ are natural numbers whose values are OS-specific constraints or chosen by the application designer.

2.4. Schedulability analysis under Posix: a recap [9]

In this paragraph, we summarize the schedulability analysis [9] of a configuration $(\mathcal{P}, \Psi_{\mathcal{P}})$ under Posix. Tasks scheduled under Posix can be described as a superposition of priority layers [9]. At each point in time, one of the ready instances with the highest priority (let's say p_i) is executed as soon as and as long as no instances in the higher priority layers (instances of tasks in $\mathcal{T}_{hp(p_i)}$) are pending. Inside each priority layer, instances are scheduled either according to FPP or RR with the restrictions that all instances belonging to the same layer have the same policy.

FPP policy is achieved when a ready instance $\tau_{i,j}$ is executed when no higher priority instances is pending. Under RR, a task τ_i has repeatedly the opportunity to execute during a time slot of maximal length $\psi_i^{\Psi_{\mathcal{P}}}$. If the task has no pending instance or less pending work than the slot is long, then the rest of the slot is lost and the task has to wait for the next cycle to resume. The time between two consecutive opportunities to execute may vary, depending on the actual demand of the others tasks, but it is bounded by $\overline{\psi}_i^{\Psi_{\mathcal{P}}} = \sum_{\tau_k \in \mathcal{T}_{p_i}^{\mathcal{P}}} \psi_k^{\Psi_{\mathcal{P}}}$ in any interval where the considered task has pending instances at any moment. In [9], worst-case response time bounds for priority layers have been derived in a way that is independent from the scheduling policies used for each layer. This analysis is based on the concept of majorizing work arrival functions, which measure a bound on the processor demand, for each task, over an interval starting at a "generalized critical instant". The majorizing work arrival function on an interval of length t for a periodic task τ_i is:

$$s_i(t) = C_i \cdot \left\lceil \frac{t}{T_i} \right\rceil . \tag{1}$$

The worst-case response time bound can be expressed as

$$\max_{j<j^*}(e_{i,j} - a_{i,j}) , \tag{2}$$

where $j^* = \min\{j \mid e_{i,j} \leq a_{i,j+1}\}$, where $a_{i,j}$ is the release of the j^{th} instance of τ_i after the critical instant and $e_{i,j}$ is a bound on the execution end of this instance. Since τ_i is a periodic task, $a_{i,j} = (j-1) \cdot T_i \; (j = 1, 2, ...)$. If τ_i is in an FPP layer, then

$$e_{i,j} = \min\{t > 0 \mid \widetilde{s}_i(t) + s_{i,j} = t\} , \tag{3}$$

where $\widetilde{s}_i(t) = \sum_{\tau_k \in \mathcal{T}_{hp(p_i)}^{\mathcal{P}}} s_k(t)$ is the demand from higher priority tasks (i.e. task in $\mathcal{T}_{hp(p_i)}^{\mathcal{P}}$) and $s_{i,j} = \sum_{i=1}^{j} C_i$ is the demand from previous instances and the current instance of τ_i. If τ_i is in an RR layer, then

$$e_{i,j} = \min\{t > 0 \mid \overline{\Psi}_i(t) + s_{i,j} = t\} , \tag{4}$$

where the demand from higher priority tasks and of all other tasks of the RR layer is**:**

$$\overline{\Psi}_i(t) = \min\left(\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_{\mathcal{P}}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_{\mathcal{P}}} - \psi_i^{\Psi_{\mathcal{P}}}) + \widetilde{s}_i(t), s_i^*(t) \right) , \tag{5}$$

where $\overline{\psi}_i^{\Psi_{\mathcal{P}}} - \psi_i^{\Psi_{\mathcal{P}}}$ is the sum of the quanta of all other tasks of the RR layer and

$$s_i^*(x) = \max_{u \geq 0} (s_i(u) + \widetilde{s}_i(u+x) + \overline{s}_i(u+x) - u) , \tag{6}$$

where $\overline{s}_i(u+x) = \sum_{\tau_k \in \mathcal{T}_{p_i}^{\mathcal{P}} \setminus \{\tau_i\}} s_k(u+x)$ is the demand from other tasks than τ_i in $\mathcal{T}_{p_i}^{\mathcal{P}}$. The algorithm for computing the worst-case response time bounds can be found in [9]. It is to stress that this schedulability analysis is sufficient but not necessary; some task sets may fail the test while they are perfectly schedulable. This will certainly induce conservative results but the approach developed here remains valid with another - better - schedulability test as long as it is sufficient and possesses the properties described in §2.5.

2.5. Scheduling under Posix 1003.1b: basic properties

Under FPP, as well as under RR, any higher priority task will preempt a lower priority task thus the following properties hold for any task τ_i :

1. all ready instances, with higher priorities than p_i, will delay the end-of-execution of the instances of τ_i. It is worth noting that this delay is not dependent on the relative priority ordering among these higher priority instances and their quantum values,
2. lower priority instances, whatever their policy, will not interfer with the execution of instances of τ_i and thus won't delay their end-of-execution.

These two properties ensure that the following lemma, which is well-known in the FPP case, holds.

Lemma 1 *[3] The worst-case response time of an instance of τ_i only depends on the set of same priority tasks, the values of their quantum and the set of higher priority tasks. The relative priority order among higher priority tasks and the values of their quantum has no influence.*

However, despite lemma 1 holding, scheduling under RR leads to scheduling anomalies. Indeed, scheduling under Posix is often counter-intuitive. For instance, it has been shown in [4], that early end-of-executions can lead to missed deadlines in configurations that would be feasible with WCETs. Similarly, removing a task with a higher priority than τ_j may lead to increased response times for τ_i (see figures 1 and 2 in [6]).

Here, we highlight that increasing the quantum size of a task can increase its response time. Figures 1 and 2 present the scheduling of task set $\mathcal{T} = \{\tau_1, \tau_2\}$ where $\tau_1 = (C_1 = 2, T_1 = 5)$ and $\tau_2 = (4, 10)$. All the tasks belong to the same layer and the chosen quantum assignments are $\Psi^{'} = \{\psi_1 = 2, \psi_2 = 2\}$ (figure 1) and $\Psi = \{\psi_1 = 2, \psi_2 = 3\}$ (figure 2) .

As it can be seen on figures 1 and 2, surprisingly the response time of τ_2 is 6 with a quantum of 2 and 8 with 3. However, with the schedulability analysis used in this study, property 1 holds and will be used to restrain the search space in section 3. A proof is given in appendix A.

Property 1 *Let τ_i be a task in a RR layer, increasing (resp. reducing) its quantum value, while reducing (resp. increasing) the quantum value of the other tasks of its RR layer, diminishes (resp. increases) the response time bound of τ_i computed with the chosen schedulability analysis.*

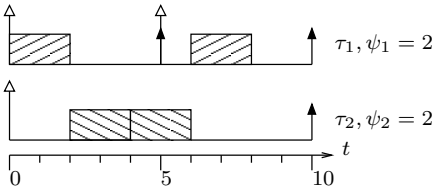

Figure 1. Scheduling of task set $\mathcal{T} = \{\tau_1, \tau_2\}$ with Round-Robin and quantum assignment $\Psi^{'} = \{\psi_1 = 2, \psi_2 = 2\}$.

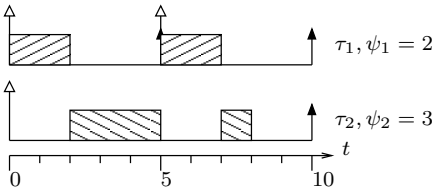

Figure 2. Scheduling of task set $\mathcal{T} = \{\tau_1, \tau_2\}$ with Round-Robin and quantum assignment $\Psi_1 = \{\psi_1 = 2, \psi_2 = 3\}$.

To be optimal, the Audsley algorithm requires that the schedulability test fulfills some properties (see §3.3). In particular, removing a task with a higher priority must not lead to increased response times. In the case of Posix 1003.1b, this imposes constraints on the schedulability test which must fulfill property 2.

Property 2 *Let τ_i be a task in RR or FPP layer, reducing its set of higher and same priority tasks, while keeping the quantum allocation unchanged within its Round-Robin layer (if τ_i is scheduled under RR), diminishes or leaves unchanged the response time bound of τ_i computed with the chosen schedulability analysis.*

It has been shown in [6] that the conservative response time bound computed with [9] ensures that property 2 holds. The proof, given in [6] in the context of a unique sysrem-wide quantum value, is still valid when different values for the quanta are possible. As it will be shown in section 3, a schedulability test which ensures that property 2 is verified, allows to use an extension of the Audsley algorithm and preserves its optimality with regards to the ability of the test to distinguish between feasible and non-feasible solutions (i.e., what is called the power of the test in the following).

3. Optimal assignment algorithm with task-specific quanta

We present here an optimal priority, scheduling policy and quanta assignment for Posix 1003.1b systems when the feasibility is assessed with schedulability analysis which verifies property 2 described in §2.5. This algorithm heavily relies on both the Audsley algorithm and the algorithm previously proposed for system-wide quantum values (called *Audsley-RR-FPP* in [6]). Here we extend previous works to the case where quanta can be chosen on a task-per-task basis, the corresponding algorithm is named the *Audsley-RR-FPP**. With the assumption made in section 2, the policy is implied by the number of tasks having the same priority level: should only one task be assigned priority level i then its policy is FPP (i.e. a RR layer of cardinality 1 is strictly equivalent to an FPP layer, see §2.1), otherwise the policy is necessarily RR. The problem is thus reduced to assigning priorities and quanta to tasks in a RR layer.

3.1. *Audsley-RR-FPP** algorithm

In the same way as the original Audsley algorithm (abridged by AA in the following), the idea is to start assigning the priorities from the lowest priority n to the highest priority 1 (line 3 in algorithm 1). The difference with AA, is that, at each priority level, the algorithm is not looking for a single task but for a set of tasks (line 5). For each such set of tasks, our algorithm examines all possible quantum assignments until it finds one suitable one.

Underlying idea. The underlying idea of the algorithm is to move , when needed, the maximum amount of workload to the lower priority levels and to schedule the tasks under RR. When an instance $\tau_{i,j}$ is assigned the same priority as $\tau_{k,h}$ and both are scheduled under RR, $\tau_{i,j}$ can delay $\tau_{k,h}$ less than if $\tau_{i,j}$ would be scheduled with a higher priority. The same argument holds for the delay induced by $\tau_{k,h}$ to $\tau_{i,j}$. Thus, as illustrated with an example in [10], where a task set that is not feasible under FPP alone, becomes feasible with RR. Of course, in the general case, combining the use of both policies is the most efficient and, as it will be shown, leads to an optimal priority and policy assignment.

Step of the algorithm. For each priority level i (line 3), the *Audsley-RR-FPP** algorithm attempts to find a schedulable subset $\mathcal{T}_i$ in subset $\mathcal{R}$ (line 5) where $\mathcal{R}$ is made of all the tasks which have not been yet assigned a priority, a policy and a quantum. The algorithm tries all possible subsets of $\mathcal{R}$, one by one, and all possible quantum assignments for each subset until a schedulable configuration is obtained or all configurations have been considered. In the latter case, the system is not schedulable (lines 7-8). Otherwise, we have found a schedulable subset, denoted by $\mathcal{T}_i$, which, in the RR case, possesses quantum assignment $\{\psi_k\}_{\tau_k \in \mathcal{T}_i}$ (lines 7 and 8). Precisely,

Input: task set $\mathcal{T} = \{\tau_1, \tau_2 ..., \tau_n\}$
Result: schedulable priority, scheduling policy and quantum assignment $\mathcal{P}_k = (\mathcal{P}, \Psi_k)$
Data: i: priority level to assign
$\mathcal{R}$: task-set with no assigned priority
$\mathcal{P}$: partial priority and policy assignment
$\Psi_{\mathcal{P}}$: partial quantum allocation

1 $\mathcal{R} = \mathcal{T}$;
2 $\mathcal{P} = \emptyset$;
3 **for** $i = n$ **to** *1* **do**
4 `try to assign priority` i`:`
5 search a schedulable subset of tasks $\mathcal{T}_i$ under quantum allocation $\{\psi_k\}_{\tau_k \in \mathcal{T}_i}$ in $\mathcal{R}$
6 **if** *no subset* $\mathcal{T}_i$ *is schedulable at priority* i **then**
7 `failure, return partial assignement:`
8 **return** $(\mathcal{P}, \Psi_{\mathcal{P}})$;
9 **else**
10 let $\mathcal{T}_i$ a schedulable subset at priority i with quantum allocation $\{\psi_k\}_{\tau_k \in \mathcal{T}_i}$;
11 `assign priority, policy and quantum:`
12 **if** $\#\mathcal{T}_i = 1$ **then**
13 $\mathcal{P} = \mathcal{P} \cup \{(\tau_k, i, sched\_fifo)\}_{\tau_k \in \mathcal{T}_i}$;
14 **else**
15 $\mathcal{P} = \mathcal{P} \cup \{(\tau_k, i, sched\_rr)\}_{\tau_k \in \mathcal{T}_i}$;
16 $\Psi_{\mathcal{P}} = \Psi_{\mathcal{P}} \cup \{\psi_k\}_{\tau_k \in \mathcal{T}_i}$;
17 **end**
18 `remove` $\mathcal{T}_i$ `from` $\mathcal{R}$`:`
19 $\mathcal{R} = \mathcal{R} \backslash \mathcal{T}_i$;
20 **end**
21 **if** $\mathcal{R} = \emptyset$ **then return** $(\mathcal{P}, \Psi_{\mathcal{P}})$;
22 **end**

Algorithm 1: *Audsley-RR-FPP** algorithm with task-specific quantum.

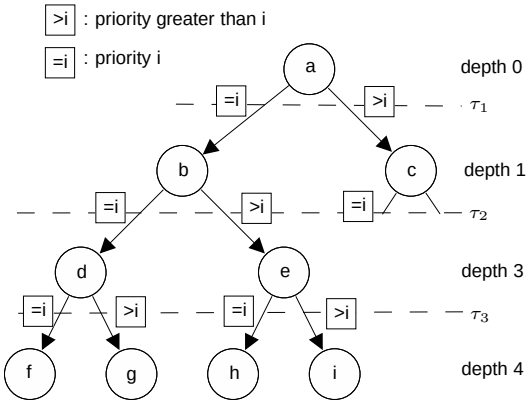

Figure 3. Search tree constructed in the search of a feasible subset of $\mathcal{R} = \{\tau_1, \tau_2, \tau_3\}$ at priority i. For instance, node b models the partial priority assignment where τ_1 is assigned priority i while node c means that τ_1 is assigned a greater priority.

$\mathcal{T}_i$ is schedulable when all tasks of $\mathcal{T}_i$ are feasible at priority i while all tasks without assignment (i.e., tasks in $\mathcal{R}\backslash\mathcal{T}_i$) have a priority greater than i. At each step, at least one task is assigned a priority and a policy (lines 11 to 17). Note that, when RR is used at least once, less than n priority levels are needed (early exit on line 21).

Looking for the set of schedulable tasks $\mathcal{T}_i$. There are $2^{\#\mathcal{R}}$ possible subsets $\mathcal{T}_i$ of $\mathcal{R}$ that can be assigned priority level i (line 5). Since the quantum can take $\|\psi\| = \psi_{\max} - \psi_{\min} + 1$ different values, there are $\|\psi\|^{\#\mathcal{T}_i}$ different quantum assignments for each subset $\mathcal{T}_i$. First, we explain the basic exhaustive tree-search used to set priorities. Then, we explain how we use a similar search to choose the quantum assignment for each possible set $\mathcal{T}_i$. A method that speeds-up the search by pruning away subtrees that cannot contain a solution is provided in §3.2.

A binary tree structure reflects the priority choices and the search for the schedulable subset is performed by exploring the tree. In the following, we call *priority-search-tree* the search tree modeling the priority choices. As an illustration, figure 3 shows the priority-search-tree corresponding to the set $\mathcal{R} = \{\tau_1, \tau_2, \tau_3\}$. Each edge is labeled either with "$= i$" (i.e., priority equal to i) or "$> i$" (i.e., priority greater than i). A label "$= i$" (resp. "$> i$") on the edge between vertices of depth k and $k+1$ means that the $(k+1)^{th}$ task of $\mathcal{R}$ belongs to the layer of priority i (resp. belongs to a layer of priority greater than i). Thus, a vertex of depth k models the choices performed for the k first tasks of $\mathcal{R}$. For instance, on figure 3, the vertex e implies that tasks τ_1 belongs to layer of priority i while task τ_2 does not. Each leaf is a complete assignment for priority level i, for instance leaf g corresponds to set $\mathcal{T}_i = \{\tau_1, \tau_2\}$.

The search is performed according to a depth-first strategy. The algorithm considers the first child of a vertex that appears and goes deeper and deeper until a leaf is reached, i.e., until the set $\mathcal{T}_i$ is fully defined. When a leaf is reached, the schedulability of $\mathcal{T}_i$ is assessed. If $\mathcal{T}_i$ is feasible, the algorithm returns, otherwise, it backtracks till the first vertex such that not all its child vertices have been explored.

To assess the schedulability of $\mathcal{T}_i$, all possible quantum assignments are successively considered. In the same manner as for the priority allocation, a tree -called *quantum-search-tree*- reflects the choices for quantum values. A depth-first strategy is used as well to explore the search space. In this case, a node has $\|\psi\|$ children where each child models a different quantum value. Here, we label the edge between vertices of depth k and $k+1$ with the quantum value of the $(k+1)^{th}$ task of $\mathcal{T}_i$. Thus, a vertex of depth k models the choices performed for the k first tasks of $\mathcal{T}_i$.

3.2. Complexity and improvements

Size of the search space. Assigning n tasks to different non-empty layers is like subdividing a set of n elements into non-empty subsets. Let k be the number of layers. The number of possible assignments is equal, by definition, to the the Stirling number of the second kind (see [1], page 824):

$$\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\left(\begin{array}{c} k \\ i \end{array}\right) i^n,$$

where $\left(\begin{array}{c} k \\ i \end{array}\right)$ is the binomial coefficient, i.e., the number of ways of picking an unordered subset of i elements in a set of k elements.

The complexity depends on the number of tasks scheduled under RR since their quantum values have to be chosen. When there are k layers, at least $n-k+1$ tasks are in an RR layer (i.e., $n-k+1$ tasks in a single RR layer and one task in each of the remaining $k-1$ FPP layers) and up to $\max(n, 2(n-k))$ (i.e., tasks are "evenly" distributed among RR layers). Since the quantum can take $\|\psi\| = \psi_{\max} - \psi_{\min} + 1$ different values, there are between $\|\psi\|^{n-k+1}$ and $\|\psi\|^{\max(n,2(n-k))}$ different quantum assignments for a configuration of k layers.

In addition, n tasks can be subdivided into $k = 1, 2, ..., n$ many layers and there are $k!$ different possible priority orderings among the k priority layers. Thus, a lower bound for the search space of the problem of assigning priority, policy and quantum for a set of n tasks is

$$\sum_{k=1}^{n}\sum_{i=0}^{k}(-1)^{(k-i)} \cdot \left(\begin{array}{c} k \\ i \end{array}\right) \cdot i^n \cdot \|\psi\|^{n-k+1}.$$

In a similar way, we derive an upper bound by replacing $\|\psi\|^{n-k+1}$ with $\|\psi\|^{\max(n,2(n-k))}$.

For instance, as can be seen on figure 4, the size of the search space comprises about $4 \cdot 10^{10}$ scheduling configurations for a set of 10 tasks. The search space grows

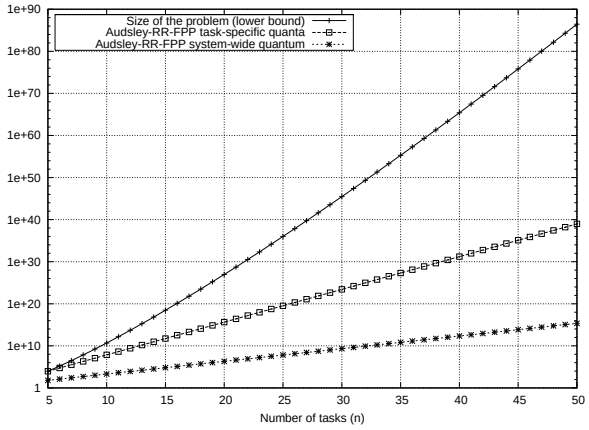

Figure 4. Complexity of the problem for a number of tasks varying from 5 to 50 when the quantum value can be chosen in the interval $[1, 5]$.

more than exponentially, thus an exhaustive search is not possible in practice in a wide range of real-time problems.

***Audsley-RR-FPP**.** Our algorithm looks at each priority level i for a subset $\mathcal{T}_i$ in $\mathcal{R}$ which is schedulable at priority i (line 5). Since at least one task is assigned to each priority level, the number of tasks belonging to $\mathcal{R}$ when dealing with priority level i is lower than or equal to i. In addition, we know that there are $\|\psi\|^k$ different quantum assignments for a subset of k tasks. Thus, at each priority level i, the algorithm examines $\sum_{j=1}^{i} \left(\begin{array}{c} i \\ j \end{array} \right) \cdot \|\psi\|^j = (\|\psi\| + 1)^i - 1$ assignments in the worst-case. Thus, for priority level from 1 to n, the algorithm considers in the worst-case a number of assignments given by:

$$\sum_{i=1}^{n} (\|\psi\| + 1)^i - 1 = \frac{1 - (\|\psi\| + 1)^{n+1}}{1 - (\|\psi\| + 1)} - (n + 1)$$

This complexity for a varying number of tasks is shown on figure 4, for instance, for a set of 10 tasks with $\psi_{min} = 1$ and $\psi_{max} = 5$ it is approximately equal to $72 \cdot 10^6$. Figure 4 shows also the size of the search space and, for comparison, the worst-case complexity of the solution proposed in [6] in the case where the quantum size is a system-wide constant. Although we achieve a great complexity reduction with regards to an exhaustive search, the complexity remains exponential in the number of tasks. Thus, in practice, our proposal is not suited for large-size task sets that would, for instance, be better handled by heuristics guiding the search towards promising parts of the search space. This is left as future work.

Complexity reduction. As seen before, the *Audsley-RR-FPP** performs an exhaustive search for each priority level. To a certain extent, it is possible to reduce the number of sets that are to be considered. Indeed, the property 3 given in this paragraph shows that it is possible to identify priority and policy assignments that are not schedulable whatever the quantum allocation. Thanks to property 2 and property 3, one can identify and prune away branches of the priority-search-tree which necessarily lead to subsets $\mathcal{T}_i$ that are not schedulable whatever the quantum assignments. Furthermore, with property 3, one can reduce in a similar manner the number of quantum assignments to consider for a particular subset $\mathcal{T}_i$ in a quantum-search-tree.

With the basic algorithm explains in §3.1, feasibility of a priority allocation is assessed at the leafs when all tasks have been given a priority by testing all quantum assignments. The idea is here to evaluate feasibility at intermediate vertices as well, by assigning a priority lower than i to the tasks for which no priority choice has been made yet. Under that configuration, if a task τ_i which is assigned the priority i is not schedulable whatever the quantum assignment, there is no need to consider the children of this vertex. Indeed, from property 2, since the priority assignment of the children of this node will increase the set of same or higher priority tasks, the response time of τ_i cannot decrease. Thus, all child vertices corresponds to priority assignments that are not schedulable. Now, it remains to identify priority and policy assignments that are not schedulable whatever the quantum allocation. The following property, proven in appendix A.3, can be stated.

Property 3 *Let $\mathcal{S}$ be a schedulability test for which property 2 holds. Let $\mathcal{T}$ be a task set and $\mathcal{P}$ be a global priority and policy assignment. Let τ_i be a task with the maximum quantum value ψ_{max} in an RR layer. Let the quantum values of all other tasks in the RR layer be set to the minimum ψ_{min}. If the response time bound of τ_i, computed with $\mathcal{S}$, is greater than its relative deadline, then, whatever the quantum assignment under $\mathcal{P}$, τ_i will remain unschedulable with $\mathcal{S}$.*

Thus, at each vertex of the priority search tree, a priority assignment $\mathcal{P}$ is not feasible whatever the quantum assignment, if a task τ_k which has a priority i is not feasible with the quantum allocation given in property 3.

Similarly, we can cut branches when exploring the quantum-search-tree of a set $\mathcal{T}_i$. The idea is again to evaluate feasibility at intermediate vertices. Since an intermediate vertex models a partial quantum assignment for a set $\mathcal{T}_i$, we assign the lowest quantum value to each task in $\mathcal{T}_i$ which has no quantum assigned yet. In that case, if a task τ_k for which the quantum has already been set at this vertex is not schedulable, then there is no need to consider the children of this vertex. Indeed, given property 1, the response time of τ_k can only increase when the the children of this vertex are considered.

The finding of this paragraph allows a very significant decrease in the average number of configurations tested by the *Audsley-RR-FPP** algorithm. For instance, for task sets constituted of 10 tasks, the algorithm examines on average only about 4000 configurations before coming up

with a feasible solution or concluding that the task set is unfeasible while it would require about $7 \cdot 10^7$ tests otherwise.

3.3. Proof of optimality

Here we show that the *Audsley-RR-FPP** algorithm is optimal in the sense that if there is a priority, policy and quantum assignment that can be identified as feasible by the schedulability analysis, it will be found by the algorithm. Let us first remind the following theorem which has been proven in [2, 3, 5, 9] for various contexts of fixed priority scheduling.

Theorem 1 *[3] Let $(\mathcal{P}, \Psi_\mathcal{P})$ be a schedulable configuration up to priority i, i.e. tasks that have been assigned the priorities from n to i are schedulable. If there exists a schedulable configuration $(\mathcal{A}, \Psi_\mathcal{A})$, then there is at least one schedulable configuration $(\mathcal{Q}, \Psi_\mathcal{Q})$ having an identical configuration as $(\mathcal{P}, \Psi_\mathcal{P})$ for priorities n to i.*

From theorem 1, we can prove the optimality of *Audsley-RR-FPP**. Indeed, if *Audsley-RR-FPP** happens to fail at level i, the priority, scheduling policy and quantum assignment $(\mathcal{P}, \Psi_\mathcal{P})$ provided by *Audsley-RR-FPP** leads to a schedulable solution up to level $i+1$. Since *Audsley-RR-FPP** performs an exhaustive search to assign level i, there cannot be any *schedulable* assignment $(\mathcal{Q}, \Psi_\mathcal{Q})$ possessing the same assignment as $(\mathcal{P}, \Psi_\mathcal{P})$ for priority $i+1$ to n. Thus, from theorem 1, there is no schedulable assignment.

We give here an intuitive proof of theorem 1, which basically is valid under Posix thanks to lemma 1 and property 2. It should be pointed out that theorem 1, and thus the optimality result of *Audsley-RR-FPP**, does not hold where property 2 is not verified by the schedulability test.

Theorem 1 holds if a schedulable configuration $(\mathcal{A}, \Psi_\mathcal{A})$ can be transformed into a schedulable configuration $(\mathcal{Q}, \Psi_\mathcal{Q})$ for which the configuration is the same as $(\mathcal{P}, \Psi_\mathcal{P})$ for priority i to n. This transformation can be done iteratively by changing the configuration of certain tasks in $(\mathcal{A}, \Psi_\mathcal{A})$ to the configuration they have in $(\mathcal{P}, \Psi_\mathcal{P})$. The procedure is the following: for priority level k from n to i, assign in $(\mathcal{A}, \Psi_\mathcal{A})$ the priority $k+n-i$ to the tasks of priority k in $(\mathcal{P}, \Psi_\mathcal{P})$ (i.e., the set $\mathcal{T}_k^\mathcal{P}$) and set their quantum value to their values $\psi_i^\mathcal{P}$ in $\Psi_\mathcal{P}$ ($\forall \tau_j \in \mathcal{T}_k^\mathcal{P}, p_j^\mathcal{A} = p_j^\mathcal{P} + n - i$, $sched_j^\mathcal{A} = sched_j^\mathcal{P}$ and $\psi_j^{\Psi_\mathcal{A}} = \psi_j^{\Psi_\mathcal{P}}$). Since at each step, tasks in $\mathcal{T}_k^\mathcal{P}$ have the same quantum assignment, the same set of higher and equal priority tasks under the current configuration $(\mathcal{A}, \Psi_\mathcal{A})$ as under $(\mathcal{P}, \Psi_\mathcal{P})$, they remain schedulable under $(\mathcal{A}, \Psi_\mathcal{A})$ by lemma 1. From property 2, the other tasks ($\mathcal{T} \setminus \mathcal{T}_k^\mathcal{P}$) meet their deadline too since the quantum assignment and the set of higher and same priority task is reduced or stay unchanged under current configuration compared to the initial configuration $(\mathcal{A}, \Psi_\mathcal{A})$. Note that in the proof the priority range has been artificially extended by adding $n-i$ lower priority levels in order to avoid the case where a higher priority tasks is moved to a non-empty layer since property 2 does not cover this situation.

4. Experimental results

Here our aim is to quantify the extent to which using task-specific quanta enables us to improve the schedulability of the system by comparison 1) with FPP and 2) with system-wide quanta.

4.1. Experimental setup.

In the following experiments, we only consider task sets that are unschedulable with FPP alone. Since we choose to consider periodic tasks with deadlines equal to periods ($D_i = T_i$), we use the Rate Monotonic priority assignment, which is optimal in that context. The global load U (i.e., $\sum_{i=1}^{n} \frac{C_i}{T_i}$) has to be necessarily greater than $n \cdot (2^{1/n} - 1)$ (from [8]) in order to be able to exhibit non-feasible task sets. In the following, we choose a quantum value of 1 for the system-wide quantum or, when task-specific quanta is considered, a quantum value which can be chosen in the interval $[1, 5]$. The actual parameters of an experiment are defined by the tuple (n, U). The utilization rate ($\frac{C_i}{T_i}$) of each task τ_i is uniformly distributed in the interval $\left[\frac{U}{n} \cdot 0.9, \frac{U}{n} \cdot 1.1\right]$ where n is the number of tasks. The computation time C_i is randomly chosen with an uniform law in the interval $[1, 30]$ and the period T_i is upper bounded by 500. The results shown on figure 5 have been obtained with 200 task sets randomly generated with the aforementioned parameters.

4.2. Schedulability improvement over FPP and system-wide quanta

Figure 5 shows the percentage of task sets that are not schedulable with FPP alone and become schedulable when using the *Audsley-RR-FPP**(task-specific quanta) and *Audsley-RR-FPP* (system-wide quanta - see [6]) algorithms that are both optimal in their context. One observes that the improvement with task-specific quanta is very important, at least 3 times better than with a system-wide quantum. When the load is lower than 84%, a solution is found in almost all cases, the percentage of successes remaining greater than 50% up to a load equal to 88%. As it was to be expected, when the load gets higher, feasible scheduling solution tends to rarefy.

Our experiments show that the combined used of RR and FPP with process-specific quanta allows to schedule a large number of task sets which are neither schedulable with FPP nor with a system-wide quantum. It is worth noting that context switch latencies were neglected while RR induces more context switches than FPP. This fact weakens to a certain extent our conclusions. A future work is to find the feasible quantum allocation that minimizes the global number of context switches.

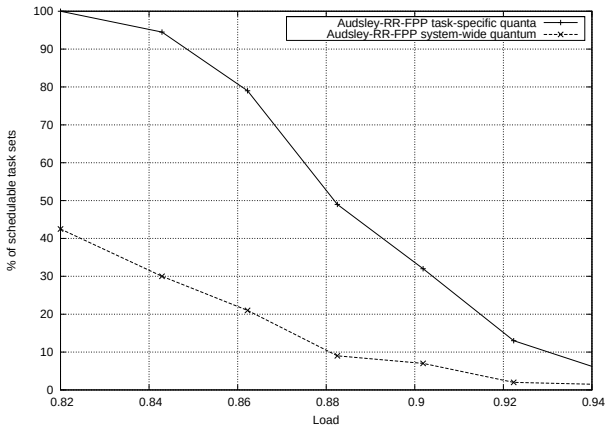

Figure 5. Percentage of task sets unschedulable with DM which become schedulable under Posix using the *Audsley-RR-FPP**(task-specific quanta) and *Audsley-RR-FPP* (system-wide quanta - see [6]) algorithms. The CPU load ranges from 0.82 to 0.94. The number of tasks is equal to 10.

5. Conclusion

In this paper, we propose a priority, policy and quantum assignment algorithm for Posix 1003.1b compliant OS that we named the *Audsley-RR-FPP**. We have shown this algorithm to be optimal in the sense that if there is a feasible schedule using FPP and RR that can be identified as such by the schedulability test, it will be found by the algorithm. A result yields by the experiments is that the combined used of FPP and RR with process-specific quanta enables to significantly improve schedulability by comparison with FPP alone and with system-wide quanta. This is particularly interesting in the context of embedded systems where the cost pressure is high, which lead us to exploit the computational resources at their fullest.

In terms of worst-case complexity, the algorithm greatly improves upon an exhaustive exploration of the search space but is still exponential in the number of tasks in the worst-case. Therefore, it is not suited to large task sets and future work is needed to develop techniques able to handle such systems. A future work is to take into account context switches and come up with a way of assigning quantum values in such a manner as to minimize the context-switch overhead.

References

[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions*. Dover Publications (ISBN 0-486-61272-4), 1970.

[2] N.C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Report YO1 5DD, Dept. of Computer Science, University of York, England, 1991.

[3] N.C. Audsley. On priority assignment in fixed priority scheduling. *Inf. Process. Lett.*, 79(1):39–44, 2001.

[4] R. Brito and N. Navet. Low-power round-robin scheduling. In *Proc. of the 12th international conference on real-time systems (RTS 2004)*, 2004.

[5] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Technical Report RR-2966, INRIA, 1996. Available at `http://www.inria.fr/rrrt/rr-2966.html` .

[6] M. Grenier and N. Navet. Scheduling configuration on Posix 1003.1b systems. Technical report, INRIA, to appear, 2007.

[7] (ISO/IEC) 9945-1:2004 and IEEE Std 1003.1, 2004 Edition. Information technology—portable operating system interface (POSIX®)—part 1: Base definitions. IEEE Standards Press, 2004.

[8] C.L. Liu and J.W Layland. Scheduling algorithms for multiprogramming in hard-real time environnement. *Journal of the ACM*, 20(1):40–61, 1973.

[9] J. Migge, A. Jean-Marie, and N. Navet. Timing analysis of compound scheduling policies : Application to Posix1003.1b. *Journal of Scheduling*, 6(5):457–482, 2003.

[10] N. Navet and J. Migge. Fine tuning the scheduling of tasks through a genetic algorithm: Application to Posix1003.1b compliant OS. *Proc. of IEEE Proceedings Software*, 150(1):13–24, 2003.

[11] K. Tindell. An extendible approach for analyzing fixed priority hard real-time tasks. Technical Report YCS-92-189, Department of Computer Science, University of York, 1992.

A. Proof of properties 1 and 2

In this appendix, we prove that the schedulability analysis [9] ensures that properties 1 and 3 hold. The first paragraph is devoted to the study of the execution end $e_{i,j}$ of $\tau_{i,j}$ computed with [9] under two configurations $(\mathcal{P}, \Psi_{\mathcal{P}})$ and $(\mathcal{P}, \Psi'_{\mathcal{P}})$ that only differ by their quantum assignment. This result is used in subsequent proofs.

A.1. Execution end bound: basic properties

We compare bounds on the execution end of τ_i under the same priority and policy assignment $\mathcal{P}$ with two different quantum allocations. Let $e_{i,j}$ and $e'_{i,j}$ be respectively the execution end bound of τ_i under $(\mathcal{P}, \Psi_{\mathcal{P}})$ and under $(\mathcal{P}, \Psi'_{\mathcal{P}})$. Since τ_i is in an RR layer, $e_{i,j}$ is computed with equation 4 of §2.4:

$$e_{i,j} = \min\{t > 0 \mid \overline{\Psi}_i(t) + s_{i,j} = t\} ,$$

where (equation 5 of §2.4)

$$\overline{\Psi}_i^{\Psi_\mathcal{P}}(t) = \min\left(\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}) + \widetilde{s}_i(t), s_i^*(x)\right),$$

where $\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}$ is the sum of the quanta of all other tasks of the RR layer. Since $s_{i,j}$, $\widetilde{s}_i(t)$ and $s_i^*(x)$ are independent of the quantum assignment (see §2.4), it is enough to compare the first term of the $\min()$ to decide which task will have the smallest response time bound. Two cases arise:

1. $\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}) > \left\lceil \frac{s_{i,j}}{\psi_i^{\Psi'_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi'_\mathcal{P}} - \psi_i^{\Psi'_\mathcal{P}})$ then we conclude $e_{i,j} \geq e'_{i,j}$,
2. otherwise:

   $$\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}) \leq \left\lceil \frac{s_{i,j}}{\psi_i^{\Psi'_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi'_\mathcal{P}} - \psi_i^{\Psi'_\mathcal{P}}),$$

   and $e_{i,j} \leq e'_{i,j}$.

When $s_i^*(x)$ is the minimum, we have $e_{i,j} = e'_{i,j}$.

From this finding we can deduce that for any other assignment $\Psi'_\mathcal{P}$, if the two following requirements are met:

requirement 1: the quantum $\psi_i^{\Psi'_\mathcal{P}}$ of τ_i in $\Psi'_\mathcal{P}$ is lower than or equal to its quantum $\psi_i^{\Psi_\mathcal{P}}$ under $\Psi_\mathcal{P}$,

requirement 2: the sum of the quanta of all other tasks of the RR layer $\mathcal{T}_i^\mathcal{P}$ under $\Psi'_\mathcal{P}$ is greater than or equal to the one under $\Psi_\mathcal{P}$, i.e., $\overline{\psi}_i^{\Psi'_\mathcal{P}} - \psi_i^{\Psi'_\mathcal{P}} \geq \overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}$ where $\overline{\psi}_i^{\Psi_\mathcal{P}} = \sum_{\tau_k \in \mathcal{T}_i} \psi_{\tau_k}^{\Psi_\mathcal{P}}$ is the sum of the quantum of all tasks of the RR layer $\mathcal{T}_i^\mathcal{P}$ under quantum allocation $\Psi_\mathcal{P}$,

then we have:

$$\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi'_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi'_\mathcal{P}} - \psi_i^{\Psi'_\mathcal{P}}) \geq \left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}), \quad (7)$$

and thus $\forall \tau_{i,j}$, $e_{i,j} \leq e'_{i,j}$ which implies that the response time bound of τ_i under $(\mathcal{P}, \Psi'_\mathcal{P})$ is greater than or equal to the response time bound under $(\mathcal{P}, \Psi_\mathcal{P})$.

A.2. Proof of property 1

Since the prerequisites of property 3 are exactly requirements 1 and 2 of §A.1, the response time bound of τ_i in property 3, is no less under $(\mathcal{P}, \Psi'_\mathcal{P})$ than under $(\mathcal{P}, \Psi_\mathcal{P})$. Since τ_i is not schedulable under $(\mathcal{P}, \Psi_\mathcal{P})$, it cannot be schedulable under $(\mathcal{P}, \Psi'_\mathcal{P})$.

A.3. Proof of property 3

We show that the bound on the execution end $e_{i,j}$ for a task in an RR layer under $\mathcal{P}$, is minimum under $\mathcal{P}$ when the quantum of τ_i is equal to ψ_{max} while the quanta of the other tasks in the layer are set to ψ_{min}. Let $\Psi_\mathcal{P}$ be the corresponding quantum assignment where

$$\left\lceil \frac{s_{i,j}}{\psi_i^{\Psi_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi_\mathcal{P}} - \psi_i^{\Psi_\mathcal{P}}) = \left\lceil \frac{s_{i,j}}{\psi_{max}} \right\rceil \cdot \Big(\sum_{\tau_k \in \mathcal{T}_{p_i}^\mathcal{P} \setminus \{\tau_i\}} \psi_{min} \Big)$$

and one notes that whatever a different quantum assignment $\Psi'_\mathcal{P}$:

$$\left\lceil \frac{s_{i,j}}{\psi_{max}} \right\rceil \cdot \sum_{\tau_k \in \mathcal{T}_{p_i}^\mathcal{P} \setminus \{\tau_i\}} \psi_{min} \leq \left\lceil \frac{s_{i,j}}{\psi_i^{\Psi'_\mathcal{P}}} \right\rceil \cdot (\overline{\psi}_i^{\Psi'_\mathcal{P}} - \psi_i^{\Psi'_\mathcal{P}})$$

since, by definition, $\psi_{max} \geq \psi_i^{\Psi'_\mathcal{P}}$ and $\psi_{min} \leq \psi_k^{\Psi'_\mathcal{P}}$. From equation 7, the execution end bound $e_{i,j}$ of $\tau_{i,j}$ is thus minimum with $\Psi_\mathcal{P}$ among the set of all possible quantum assignments.

Notations

- $\mathcal{T} = \{\tau_1, ..., \tau_n\}$: a set of n periodic tasks
- $\mathcal{P}$: priority and policy assignment
- $\Psi_\mathcal{P}$: a specific quantum allocation under assignment $\mathcal{P}$
- $(\mathcal{P}, \Psi_\mathcal{P})$: a priority, policy and a quantum assignment
- $\mathcal{T}_i^\mathcal{P}$: subset of tasks assigned to priority level i under $\mathcal{P}$
- $\mathcal{T}_{hp(i)}^\mathcal{P}$: subset of tasks assigned to a higher priority than i under $\mathcal{P}$
- $\mathcal{T}_{lp(i)}^\mathcal{P}$: subset of tasks assigned to a lower priority than i under $\mathcal{P}$
- $\psi_i^{\Psi_\mathcal{P}}$: Round-Robin quantum for task τ_i under $\Psi_\mathcal{P}$
- $\overline{\psi}_i^{\Psi_\mathcal{P}}$: sum of the quanta of all tasks in layer $\mathcal{T}_i$ under $\Psi_\mathcal{P}$
- $s_i(t) = C_i \cdot \left\lceil \frac{t}{T_i} \right\rceil$: majorizing work arrival function on an interval of length t for a periodic task τ_i
- $\widetilde{s}_i(t) = \sum_{\tau_k \in \mathcal{T}_{hp(p_i)}^\mathcal{P}} s_k(t)$: the demand from higher priority tasks under $\mathcal{P}$
- $s_{i,j} = \sum_{i=1}^{j} C_i$: the demand from previous instances plus demand of current instance $\tau_{i,j}$ of τ_i
- $\overline{s}_i(x) = \sum_{\tau_k \in \mathcal{T}_{p_i}^\mathcal{P} \setminus \{\tau_i\}} s_k(x)$ is the demand from all other tasks than τ_i at priority level i under assignment $\mathcal{P}$.