PFair scheduling of late released tasks with constrained deadlines

Sadouanouan Malo[1] and Annie Choquet-Geniet[2] and Moustapha Bikiengar[3]

[1]Polytechnic University of Bobo Dioulasso.Information Technology High School. 01 BP 1091 Bobo Dioulasso 01, Burkina Faso.
[2]University of Poitiers. Laboratory of Applied Computer Science. 1 Av. Clément Ader BP 40109 - 86961 Futuroscope Chasseneuil-France.
[3]University of Koudougou. 01 BP 376 Koudougou, Burkina Faso.
E-mail: [1(*)]sadouanouan.malo@ensma.fr, [2]annie.geniet@univ-poitiers.fr, [2]bmoustaph@yahoo.fr

[*]Corresponding author

ABSTRACT. Pfair scheduling has usually been applied in the context of synchronous periodic task systems with implicit deadlines. This paper addresses the problem of scheduling asynchronous hard real-time tasks with constrained deadlines using a Pfair strategy on multiprocessor systems. First, we extend the notion of Pfairness to the context of asynchronous tasks with constrained deadline. Then we investigate feasibility conditions, we propose a rather efficient one and we illustrate the relevance of our criteria through some simulations.

RÉSUMÉ. Les ordonnancements P-équitables ont jusque là été étudiés pour des systèmes de tâches à départs simultanés et à échéances sur requêtes. Notre objectif dans ce papier est d'étendre la définition de la P-équité aux systèmes de tâches à départs différés et à échéances contraintes puis de dégager une condition suffisante d'ordonnançabilité. Enfin, des simulations que nous avons effectuées ont permis d'illustrer la pertinence de nos résultats et de mesurer l'impact de la variation des facteurs d'utilisation et de charge du système sur l'ordonnaçabilité du système.

KEYWORDS : Scheduling - Fairness - Pfair feasibility

MOTS-CLÉS : Ordonnancement - Équité - Ordonnançabilité P-équitable

1. Introduction

The temporal validation relies on the definition of a scheduling policy and on the proof that the temporal requirements are all met. This issue is rather well mastered for uniprocessor systems, but for more complex systems, many open problems still deserve to be investigated. We here consider multiprocessor platforms, composed of identical processors, and the global scheduling of hard real-time tasks. We assume that preemption and interprocessor migration are permitted and parallelism forbidden (at any time, a task can run on at most one processor). In this context, it has been shown that no online scheduling algorithm can be optimal [8][5]. In their paper, [3][4] proved that the problem of optimally scheduling synchronous periodic tasks with implicit deadlines on identical multiprocessor platforms could be solved at run- time in polynomial time using Pfair scheduling algorithms. Pfair scheduling algorithms have been widely investigated last years [1][2][10]. In all these works, the notion of Pfairness is defined in the context of synchronous tasks with implicit deadlines. In [2], asynchronous systems are considered, where asynchronism means that some of the first task slots do not take place. But the first job of any task is still assumed to be released at time 0. In [1], sporadic tasks are considered: periods correspond only to the minimum elapsed time between two consecutive releases. We consider here a slightly different notion of asynchronism: the first release dates of the different tasks are no more assumed to be equal. But all the task slots are assumed to occur. The problem of scheduling tasks with arbitrary deadlines in a Pfair way on multiprocessors has been addressed in [9] and a static-priority scheme to schedule a set of such tasks has been presented. In this paper we investigate the extension of Pfairness in order to design Pfair scheduling schemes for asynchronous tasks with deadlines less than or equal to periods. We first extend and adapt Pfairness definition to this new context and then we propose a a sufficient but rather effective feasibility condition and then we present some simulation results.

The remainder of the paper is organized as follows. In section 2, we formally define the Pfair multiprocessor scheduling problem. In section 3, we prove that Pfair algorithm exists for any periodic task set and we give sufficient feasibility conditions. In section 4, we present the simulations results. Conclusions and perspectives are given in section 5.

2. Pfair scheduling

We adopt the following notations: for any real number x, $\lfloor x \rfloor$ is the integer immediately below or equal to x and $\lceil x \rceil$ is the integer immediately above or equal to x. A slot t denotes the time interval $[t, t+1)$. We assume that processors are allocated for integral number of slots, thus a task cannot be preempted within a slot. We consider a platform composed of m identical processors. We use the classical task model.

Definition 1. *A task $\tau_i = \langle r_i, C_i, D_i, T_i \rangle$ is characterized by four parameters: its first release time r_i, its per period worst case execution time C_i, its relative deadline D_i and its period T_i. The weight of task τ_i is $U_i = \frac{C_i}{T_i}$ and its density is $CH_i = \frac{Ci}{D_i}$.*

We have $r_i \geq 0$ and we assume that C_i, D_i and T_i are integral and verify $C_i \leq Di \leq T_i$. Thus we have $0 < U_i \leq 1$. In this section we consider a set Γ of tasks such that $r_i = 0$ (the tasks are said to be synchronous) and $D_i = T_i$.

Definition 2. *A schedule for a task set* Γ *is a function* $S : \Gamma \times \mathbb{N} \rightarrow \{0, 1\}$*, such that* $\forall t \in \mathbb{N} : \sum_{\tau_i \in \Gamma} S(\tau_i, t) \leq m$ *with* $S(\tau_i, t) = 1$ *if task* τ_i *is scheduled in slot* $[t, t+1)$ *and* $S(\tau_i, t) = 0$ *otherwise. A schedule is then valid iff* $\forall \tau_i \in \tau$, $\sum_{t=0}^{t=r_i-1} S(\tau_i, t) = 0$ *and*

$$\forall k \in \mathbb{N}^*, \sum_{t=0}^{t=r_i+(k-1)T_i+D_i-1} S(\tau_i, t) = \sum_{t=0}^{t=r_i+kT_i-1} S(\tau_i, t) = kC_i$$

A fair schedule is approximately an ideal fluid schedule such that, at any time t, each task has been processed for $\omega_i(t) = U_i \times t$ processor time units. Now, since processor time is allocated in integral number of slots, the ideal behaviour is approximated by either the integer directly above or directly beyond. A schedule S is said to be **Pfair** iff $\forall \tau_i, t : \tau_i \in \Gamma, t \in \mathbb{N}, -1 < \omega_i(t) - \sum_{j=0}^{j=t-1} S(\tau_i, t) < 1$. Informally, the allocation error associated with each task must always be less than one slot. A Pfair algorithm has the following high-level structure: at each time $t \geq 0$, a dynamic priority is assigned to each task and the m highest-priority tasks are scheduled in slot t. It can be summarized as follows [4]:

1) All **urgent** tasks are scheduled, where a task is urgent at time t if it must be scheduled at time t either the fairness condition would be violated.

2) **Contending** tasks are sorted, where a task is contending if the fairness condition is violated neither if it is processed nor if it isn't.

3) The remaining processors are allocated to the highest-priority contending tasks. Three Pfair scheduling algorithms, PF, PD and PD^2 $(4; 3; 1; 2)$, are known to be optimal on an arbitrary number of processors as stated in the theorem 1. Furthermore, there exists a sufficient and necessary feasibility feasibility condition in the context of synchronous independent tasks with implicit deadlines. The three algorithms differ in the choice of tie-breaking rules. In the remainder of the paper, we will consider the algorithm PF to illustrate our results.

Theorem 1. *[4][2] The algorithms* PF*,* PD *and* PD^2 *are optimal for sets of synchronous independent tasks with implicit deadlines. Moreover, such a task set* Γ *has a Pfair schedule on m processors if and only if* $\sum_{\tau_i \in \Gamma} \frac{C_i}{T_i} \leq m$*.*

3. Extension of Pfairness

Our aim is now to prove that fairness can be extended to all independent periodic task sets. We thus consider a periodic task set Γ such that $\forall \tau_i \in \Gamma$, $\tau_i = \langle r_i, C_i, D_i, T_i \rangle$ with $r_i \geq 0$ and $D_i \leq T_i$. We first extend the notion of Pfairnes in the following way: in an ideal schedule of a periodic task set, each task τ_i must have received at time t, $\omega_i(t)$ processor time units. $\omega_i(t)$ is such that:

$$\omega_i(t) = \begin{cases} t \in [0, r_i) \Rightarrow \omega_i(t) = 0 \\ t \in [kT_i + r_i, kT_i + D_i + r_i) \Rightarrow \omega_i = kC_i + \frac{C_i}{D_i}(t - kT_i - r_i) \\ t \in [kT_i + D_i + r_i, (k+1)T_i + r_i) \Rightarrow \omega_i = (k+1)C_i \end{cases}$$

where $k = \left\lfloor \frac{t}{T_i} \right\rfloor$ represents the instance number of the pending instance of the task.

We first have carried out some simulations using the algorithm PF. These simulations lead to two constatations: firstly, if deadlines are less than or equal to periods, if $\sum_{i=1}^{i=n} \frac{C_i}{D_i} \leq m$, the task set is feasible on m processors and secondly if asynchronous tasks with implicit deadlines are considered, Baruah's condition still holds. This leads us to infer the following result:

Theorem 2. *Given a periodic task set* Γ*, if* $\sum_{i=1}^{i=n} \frac{C_i}{D_i} \leq m$*, then* Γ *has a valid Pfair schedule on* m *processors over any time interval* $[0, t)$*.*

The proof of the theorem is an adaptation of the proof presented in [4] to prove theorem 1 for PF. It is based on the graph theory. We prove that a Pfair schedule exists on any time interval $[0, L)$. In the further, $CTR_{\tau_i}(t)$ denotes the complete processor demand of all sub-tasks of task τ_i whose feasibility intervals are included in $[0, t]$. Thus, $CTR_{\tau_i}(t)$ is equal to j where j is such that $d_i^j \leq t < d_i^{j+1}$. We first define a weighted digraph G and prove that if G has an integral flow of size $\sum_{\tau_i \in \Gamma} CTR_{\tau_i}(L)$ then the task set Γ has a Pfair schedule.

Definition 3. *The Pfair-graph is the weighted digraph* $G(L)$ *is defined as* $G(L) = (V, E)$ *with:* $V = V_0 \cup V1_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5$ *and* $E = E_1 \cup E_2 \cup E_3 \cup E_4$*;* $\mathbf{V}_0 = \{(Source\rangle\}$*;* $\mathbf{V}_1 = \{\langle 1, \tau_i\rangle, \tau_i \in \Gamma\}$*;* $\mathbf{V}_2 = \{\langle 2, \tau_{i0}, 0\rangle, \tau_i, i = 1 \ldots n \text{ s.t. } r_i > 0\} \cup \left\{\langle 2, \tau_{i0}, j\rangle, (i, j) \text{ s.t. } i = 1 \ldots n, j \in \left[1, \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\right)\right\}$*;*
$\mathbf{V}_3 = \{\langle 3, \tau_{i0}, t\rangle, (i, t) \text{ s.t. } i = 1 \ldots n, t \in [0, r_i)\} \cup \{\langle 3, \tau_i, t\rangle, (i, t) \text{ s.t. } i = 1 \ldots n, t \in [kT_i + r_i, tT_i + r_i + D_i), with 1 \leq k \leq \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\} \cup \left\{\langle 3, \tau_{i0}, t\rangle, (i, t) \text{ s.t. } i = 1 \ldots n, t \in [kT_i + D_i, (k+1)T_i), with 1 \leq k \leq \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\right\}$*;*
$\mathbf{V}_4 = \{\langle 4, t\rangle, t \in [0, L)\}$ *and* $\mathbf{V}_5 = \{\langle sink\rangle\}$*. Edges and capacities are defined by:*
$\mathbf{E}_0 = \{(\langle source\rangle, \langle 1, \tau_i\rangle, CTR_{\tau_i}(L)), i = 1 \ldots n\}$*;* $\mathbf{E}_1 = \{(\langle 1, \tau_i\rangle, \langle 2, \tau_{i0}, 0\rangle), i = 1 \ldots n \text{ s.t. } r_i > 0\}$*;* $\mathbf{E}_2 = \{(\langle 2, \tau_{i0}, 0\rangle, \langle 3, \tau_{i0}, t\rangle, 0), (i, t) \text{ s.t. } i = 1 \ldots n, t \in [0, r_i)\} \cup \left\{\left(\left\langle 2, \tau_i^j\right\rangle, \langle 3, \tau_i, t\rangle, 1\right), (i, j, t) \text{ s.t. } i = 1 \ldots n, j \in [0, CTR_{\tau_i}(L)), t \in \left[r_i^j, d_i^j\right)\right\} \cup \left\{(\langle 2, \tau_{i0}, j\rangle, \langle 3, \tau_i, t\rangle, 0), (i, j, t) \text{ s.t. } i = 1 \ldots n, j \in \left[1, \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\right), t \in [kT_i + r_i + D_i, (k+1)T_i + r_i) with 1 \leq k \leq \left\lfloor \frac{L - r_i}{T_i} \right\rfloor t\right\}$*;*
$\mathbf{E}_3 = \{(\langle 3, \tau_{i0}, t\rangle, \langle 4, t\rangle, 0), t \in [0, r_i)\} \cup \left\{(\langle 3, \tau_i, t\rangle, \langle 4, t\rangle, 1), t \in [kT_i + r_i, kT_i + r_i + D_i) with 1 \leq k \leq \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\right\} \cup \{(\langle 3, \tau_{i0}, t\rangle, \langle 4, t\rangle, 0), (i, t) \text{ s.t. } i = 1 \ldots n, t \in [kT_i + r_i + D_i, (k+1)T_i) with 1 \leq k \leq \left\lfloor \frac{L - r_i}{T_i} \right\rfloor\}$*;* $\mathbf{E}_4 = \{(\langle 4, t\rangle, \langle sink\rangle, m), t \in [0, L)\}$.

In order to prove the theorem, we first establish the following lemma.

Lemma 1. *If the Pfair-graph* $G(L)$ *has an integral flow of size* $\sum_{\tau_i \in \Gamma} CTR_{\tau_i}(L)$*, then* Γ *has a Pfair schedule on* $[0, L]$*.*

Proof. Let us assume that such an integral flow exists. We first define a schedule S deduced from the Pfair -graph as:

Definition 4. *Let f be an integral flow of size $\sum_{\tau_i \in \Gamma} CTR_{\tau_i}(L)$ of G, we define SG as follows. For $\tau_i \in \Gamma$, $t \in \mathbb{N}$,*

$$SG(\tau_i, t) = \begin{cases} 1 \text{ if } t \in [0, L) \wedge \left(\exists j \in [0, CTR_{\tau_i}(L)) \text{ s.t. } f\left(\left\langle 2, \tau_i^j \right\rangle, \langle 3, \tau_i, t\rangle, 1\right) = 1\right) \\ 0 \text{ otherwise} \end{cases}$$

We show that SG is Pfair over the time interval $[0, L)$.

The size of the flow is $\sum_{i=1}^{i=n} CTR_{\tau_i}(L)$, thus each node of V_1 is filled to capacity, i.e. the flow carried by each link from source to $\langle 1, \tau_i\rangle$ is equal to $CTR_{\tau_i}(L)$. Each node in V_1 has exactly $CTR_{\tau_i}(L)$ outgoing edges of capacity 1, the other outgoing edges have a null capacity thus they receive a flow equal to 0. Thus each node $\left\langle 2, \tau_i^j \right\rangle$ of V_2 receives a flow equal to 1. Then, each node $\left\langle 2, \tau_i^j \right\rangle$ has one single outgoing edge which carries a flow equal to 1, the other outgoing edges carry a flow equal to 0. Now, each node of V_3 has exactly one outgoing edge. This edge carries a flow equal to the flow carried by its incoming edge. In the same way, the nodes in V_4 have one single outgoing edge, which carries the cumulated flow carried by their incoming edges. Since the capacity of this outgoing edge is m, at most m incoming edges carry a flow equal to 1. Thus at most m sub-tasks are processed. Now, the potential exists for a task τ_i to get scheduled twice at the same time (if $r_i^j = d_i^{j-1} - 1$). But since the edge from $\langle 3, \tau_i, t\rangle$ to $\langle 4, t\rangle$ has a capacity equal to 1, this situation is avoided: if this edge carries a flow equal to 1, only one incoming edge carries a non-null flow. Thus two different processed sub-tasks cannot belong to the same task. Thus at any time t in $[0, L)$, there exists at most m tasks such that $SG(\tau_i, t) = 1$. Furthermore, $f\left(\left(\left\langle 2, \tau_i^j \right\rangle, \langle 3, \tau_i, t\rangle, 1\right)\right) = 1$ implies that $r_i^j \leq t < d_i^j$ thus, each processed sub-task is processed in its feasibility window. Finally, each sub-task is effectively processed. Indeed, there exists $\sum_{i=1}^{i=n} CTR_{\tau_i}(L)$ sub-tasks in $[0, L)$. Following the definition of SG, the number of processed sub-tasks is equal to the global incoming flow of vertices of V_3. And the global input flow of V_3 is constant, equal to $\sum_{i=1}^{i=n} CTR_{\tau_i}(L)$ by definition of the flow. Thus each sub-task is processed. The schedule SG is thus Pfair on $[0, L)$. □

We now prove the existence of an integer flow of size $\sum_{i=1}^{i=n} CTR_{\tau_i}(L)$. We use the following flow assignments:

Definition 5. *Let f be the flow defined as:*

$f(\langle source\rangle, \langle 1, \tau_i\rangle, (L)) = CTR_{\tau_i}(L)$, $f(\langle 1, \tau_i\rangle, \langle 2, \tau_{i0}, 0\rangle, 0) = 0$,

$f(\langle 1, \tau_i\rangle, \langle 2, \tau_i, j\rangle, 1) = 1$, $f(\langle 1, \tau_i\rangle, \langle 2, \tau_{i0}, j\rangle, 0) = 0$,

$f(\langle 2, \tau_{i0}, 0\rangle, \langle 3, \tau_{i0}, t\rangle, 0) = 0$, $f(\langle 2, \tau_{i0}, j\rangle, \langle 3, \tau_{i0}, t\rangle, 0) = 0$,

$f\left(\left\langle 2, \tau_i^j \right\rangle, \left\langle 3, \tau_i, r_i^j \right\rangle, 1\right) = CH_i - j + r_i^j CH_i$

$d_i^j - 1 = r_i^{j+1} \Rightarrow f\left(\left\langle 2, \tau_i^j \right\rangle, \left\langle 3, \tau_i, d_i^j - 1 \right\rangle, 1\right) = j + 1 - r_i^{j+1} CH_i$

$otherwise \Rightarrow f\left(\left\langle 2, \tau_i^j \right\rangle, \langle 3, \tau_i, t\rangle, 1\right) = CH_i$

$f(\langle 3, \tau_{i0}, t\rangle, \langle 4, t\rangle, 0) = 0$, $f(\langle 3, \tau_{i0}, t\rangle, \langle 4, t\rangle, 1) = CH_i$,

$f\left(\langle 4,t\rangle,\langle sink\rangle,m\right)=\sum_{\substack{\tau_i\in\Gamma\\ r_i+kT_i\leq t>r_i+kT_i+D_i}} CH_i$

Lemma 2. *f is a flow of size $\sum_{\tau_i\in\Gamma} CTR_{\tau_i}(L)$ of $G(L)$.*

Proof. We first prove that the capacity constraints are met. Edges in E_0, E_1 are filled to capacity, and edges in E_3 carry flows either equal to 0 or to the density CH_i which is less than or equal to 1, thus capacity constraints are met. If an edge in E_4 is considered, it carries a flow $\sum_{\substack{\tau_i\in\Gamma\\ r_i+kT_i\leq t<r_i+kT_i+D_i}} CH_i \leq \sum_{\tau_i\in\Gamma} CH_i$ now by assumption we have $\sum_{\tau_i\in\Gamma} CH_i \leq m$, so the capacity constraint is met. Finally, for edges in E_2, we must prove that $CH_i - \left(jr_i^j CH_i\right) \leq 1$ and $(j+1) - r_i^{j+1}CH_i \leq 1$ if $d_i^j - 1 = r_i^{j+1}$. We have $r_i^j = \left\lfloor \frac{j}{CH_i} \right\rfloor$ thus $\frac{j}{CH_i} - 1 < r_i^j \leq \frac{j}{CH_i}$ thus $CH_i < r_i^j CH_i - j \leq 0$ thus $0 < CH_i - \left(j - r_i^j CH_i\right) \leq CH_i \leq 1$. We prove that $(j+1) - r_i^{j+1}CH_i \leq 1$ using similar arguments. Thus capacity constraints are all met. We must then show that the flow is preserved at every inner vertex. For null capacity nodes, the flow is clearly preserved. For any node $\langle 1,\tau_i\rangle$ of V_1, the incoming flow is $CTR_{\tau_i}(L)$, and the outgoing flow is equal to the number of sub-tasks since edges are filled to capacity, thus the outgoing flow is $CTR_{\tau_i}(L)$ too. Each vertex $\langle 2,\tau_i,j\rangle$ has an incoming flow of 1. Each vertex $\langle 2,\tau_i,j\rangle$ has $d_i^j - 1 - r_i^j$ outgoing edges. Then the flow out of $\langle 2,\tau_i,j\rangle$ is, if $d_i^j - 1 = r_i^{j+1}$, $CH_i - \left(j - r_i^j CH_i\right) + CH_i\left(d_i^j - r_i^j - 2\right) + (j+1) - r_i^{j+1}CH_i$ which simplifies to 1. Otherwise, we have $d_i^j - 1 \neq r_i^{j+1}$ thus $\left\lceil \frac{j+1}{CH_i} \right\rceil - 1 \neq \left\lfloor \frac{j+1}{CH_i} \right\rfloor$ which means that $\frac{j+1}{CH_i}$ is integral thus $d_i^j = \frac{j+1}{CH_i}$. Now, the flow out is $CH_i - \left(j - r_i^j CH_i\right) + CH_i\left(d_i^j - r_i^j - 1\right)$ which then simplifies to 1. There is only one outgoing edge leaving any vertex $\langle 3,\tau_i,t\rangle$ of V_3, which carries a flow equal to CH_i. If $d_i^j - 1 = r_i^{j+1}$, then there are two incoming edges which carry a flow of size $(j+1) - r_i^{j+1}CH_i + CH_i - \left((j+1) - r_i^{j+1}CH_i\right) = CH_i$. Otherwise there is only one incoming edge which carries a flow equal to CH_i. We consider finally a vertex $\langle 4,t\rangle$ of V_4. Its incoming edges with non zero capacity are edges $(\langle 3,\tau_i,t\rangle,\langle 4,t\rangle,1)$ with $r_i + kT_i \leq t < ri_i + kT_i + D_i$. Thus the incoming flow is $\sum_{\substack{\tau_i\in\Gamma\\ r_i+kT_i\leq t>r_i+kT_i+D_i}} CH_i$, which is thus equal by definition to the flow of the unique outgoing edge. Thus, we proved that the flow is preserved at any inner node. Thus f is a flow of size $\sum_{\tau_i\in\Gamma} CTR_{\tau_i}(L)$. □

Now Lemma 2 implies the existence of a fractional flow of size $\sum_{\tau_i\in\Gamma} CTR_{\tau_i}(L)$ for the Pfair-graph $G(L) = (V,E)$. Since capacities are integral, this implies the existence of an integral flow of size $\sum_{\tau_i\in\Gamma} CTR_{\tau_i}(L)$ in $G(L)$ [6]. Then Lemma 1 proves that a Pfair schedule can be constructed. This proves the theorem 2.

Then we extend the algorithm PF to periodic task sets ($r_i \leq 0$, $D_i \leq Ti$). Here, a task can be *Urgent*, *Tnegru* or *Contending* if $t \in [kT_i + r_i, kT_i + r_i + D_i)$ and is *Idle* if $t < r_i$ or $t \in [kT_i + r_i + D_i, (k+1)Ti_i)$. The extension is then straightforward: at each time t, *Urgent* tasks are scheduled, *Contending* tasks are sorted and the firsts of them are allocated to the remaining processors.

4. Simulation results

The next point of interest is to determine whether this condition is efficient. We thus investigate the soundness of our bound. For that purpose, we have carried out some simulations. We have first implemented a task set simulator and a scheduler based on our extension of PF. We have then generated a significant number of task sets with different values of either $U = \sum_{i=1}^{i=n} \frac{C_i}{T_i}$ or $CH = \sum_{i=1}^{i=n} \frac{C_i}{D_i}$. In order to limit the scheduling step, we have generated periods according to Goossens methodology [7], which permits to have a bound for the hyperperiod (the LCM of the task periods). For our simulations the upper bound of hyperperiods is set to 210. Offsets, constrained deadlines and WCET are chosen uniformly within respectively the intervals $[0, T_i]$, $[1, T_i]$ and $[1, D_i - 1]$. Then we have scheduled them with our adapted PF policy, in order to estimate the ratio of feasible sets among them. All simulations have been carried out over a time interval included in $\left[0, \underset{i\in[1,n]}{max}\,(r_i) + 2T\right)$. Then we have considered several cases. Task sets can be synchronous or asynchronous, with implicit or constrained deadlines. They are characterized by either $U \leq m$ or $CH \leq m$ or $CH > m$. For each case, we have generated a sample of 5000 tasks sets for simulations. As expected, we find a Pfair feasibility rate of 100% for systems with implicit deadlines: for synchronous systems, it corresponds to Baruahs theorem (theorem 1), and for asynchronous systems, it comes from our result (theorem 2). For the other cases, we conclude that for constrained task sets, $U \leq m$ is no more a sufficient condition, since there exists Pfair unfeasible task sets with a utilization factor less than m and $CH \leq m$ is not a necessary condition since there exist Pfair feasible task sets with $CH > m$.

Then we refined our simulations in order to determine the incidence of U or CH on Pfair feasibility. We have considered systems m processors. For each case, we generate samples for different values of CH between m and $m + 1$, namely $CH = m + \frac{k}{10}$ ($0 \leq k \leq 10$). For each value of CH, we again determine the ratio of Pfair feasible task sets. We have seen that this rate decreases rather quickly when CH increases. If CH remains close to m, the rate of valid system remains high, but the slope of the curve is high and consequently the rate becomes very small if CH approaches $m + 1$. We can conclude from these results that our bound is rather good in the sense that only few systems rejected by our test are in fact Pfair feasible.

We also investigated the correlation between U and the Pfair feasibility. We consider systems with m processors, and synchronous task systems with constraint deadlines. We see that if U is close to m, then quite no systems are Pfair feasible. But if U is less than $\frac{m}{2}$, we have 100% of Pfair feasible systems. Further investigations must be done here.

5. Conclusions

We have considered real-time applications running on a multiprocessor platform. For such systems, we have extended the notion of Pfairness to any set of periodic independent task set with implicit or constrained deadlines. We have considered as well late released tasks as constraint deadlines. We have proposed a sufficient condition and given an adapted version of PF. We proved the existence of a Pfair schedule if $CH = \sum_{i=1}^{i=n} \frac{C_i}{D_i} \leq$

m. Then we have presented simulation results, which illustrate the soundness of our sufficient conditions. These simulations have shown that if CH increases from m, the rate of Pfair feasible systems decreases quickly. We also present some results about the incidence of the utilisation factor $U = \sum_{i=1}^{i=n} \frac{C_i}{T_i}$. We have speculated about the existence of a lower U, which can be used for any periodic task set with implicit or constrained deadlines.

6. References

[1] J. Anderson, A. Block, and A. Srinivasan. Pfair scheduling : Beyond periodic task sytems. In *Proceedings of the* 12^{th} *Euromicro Conference on Real-Time Systems*, pages 35–43. Chapman and Hall, 2000.

[2] J. Anderson, P. Holman, and A. Srinivasan. Fair scheduling of real time tasks on multiprocessors. *Handbook of scheduling : Algorithms, Models and Performance analysis*, pages 31.1–31.21, 2004.

[3] S. Baruah, J. Gehrke, and C.G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the* 9^{th} *International Parallel Processing Symposium*, pages 280–288, April 1995.

[4] S.K. Baruah, N.K. Cohen, C.G. Plaxton, and D.A. Varvel. Proportionate progress : a notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.

[5] M.L. Dertouzos and A.K.L. Mok. Multiprocessor scheduling in hard real-time environment. *IEEE transactions on sofware Engineering*, 15(12):1497–1506, 1989.

[6] L.R. Ford Jr and D.R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[7] C. Macq and J. Goossens. Limitation of the hyper-period in real-time periodic task set generation. In Teknea, editor, *Proceedings of the 9th international conference on real-time systems*, pages 133–148, Paris France, March 2001. ISBN 2-87717-078-0.

[8] A.K. Mok and M.L. Dertouzos. Multi processor scheduling in a hard real-time environment. In *Proc. of* 7^{th} *Texas Conference on Computer Systems*, 1978.

[9] S. Ramamurthy. Scheduling periodic hard real-time tasks with arbitrary deadlines on multiprocessors. In *IEEE Real-Time Systems Symposium*, 2002.

[10] A. Srinivasan, P. Holman, and J.H. Anderson. Integrating aperiodic and recurrent tasks on fair- scheduled multiprocessors. In *14th Euromicro Conference on Real- Time Systems*, pages 189–198, Vienna, Austria, June 2002. IEEE Computer Society.