Feasibility Intervals for Multiprocessor Fixed-Priority Scheduling of Arbitrary Deadline Periodic Systems

Liliana Cucu* Joël Goossens

Université Libre de Bruxelles, C.P. 212
50 Avenue Franklin D. Roosevelt
1050 Brussels, Belgium
E-mail: {liliana.cucu, joel.goossens}@ulb.ac.be

Abstract

In this paper we study the global scheduling of periodic task systems with arbitrary deadlines upon identical multiprocessor platforms. We first show two very general properties which are well-known for uniprocessor platforms and which remain for multiprocessor platforms: (i) under few and not so restrictive assumptions, we show that any feasible schedule of arbitrary deadline periodic task systems is periodic from some point and (ii) for the specific case of synchronous periodic task systems, we show that the schedule repeats from the origin. We then present our main result: any feasible schedule of asynchronous periodic task sets using a fixed-priority scheduler is periodic from a specific point. Moreover, we characterize that point and we provide a feasibility interval for those systems.

1 Introduction

The use of computers to control safety-critical real-time functions has increased rapidly over the past few years. As a consequence, real-time systems — computer systems where the correctness of each computation depends on both the logical results of the computation and the time at which these results are produced — have become the focus of much study. Since the concept of "time" is of such importance in real-time application systems, and since these systems typically involve the sharing of one or more resources among various contending processes, the concept of scheduling is integral to real-time system design and analysis. Scheduling theory as it pertains to a finite set of requests for resources is a well-researched topic. However, requests in real-time environment are often of a recurring nature. Such systems are typically modelled as finite collections of simple, highly repetitive tasks, each of which generates jobs in a very predictable manner. These jobs have upper bounds upon their worst-case execution requirements, and associated deadlines. In this work, we consider periodic task systems, each periodic task τ_i generates jobs at each integer multiple of its period T_i and the jobs must be executed for at most C_i time units and completed by its relative deadline D_i. The first job of a task τ_i is released at time O_i (the task offset). If there is a time instant at which jobs of all tasks are released synchronously, the system is said to be *synchronous*; otherwise the system is said to be *asynchronous*. We shall distinguish between *implicit deadline* systems where $D_i = T_i, \forall i$; *constrained deadline* systems where $D_i \leq T_i, \forall i$ and *arbitrary deadline* systems where there is no constraint between the deadline and the period.

The *scheduling algorithm* determines which job[s] should be executed at each time instant. When there is at least one schedule satisfying all constraints of the system, the system is said to be *feasible*. More formal definitions of these notions are given in Section 2.

Uniprocessor real-time systems are well studied since the seminal paper of Liu and Layland [1] which introduces a model of constrained deadline systems. The arbitrary deadline systems model is firstly considered in [2].

The literature considering scheduling algorithms and feasibility tests for uniprocessor scheduling is tremendous. In contrast for *multiprocessor* parallel machines the problem of meeting timing constraints is a relatively new research area.

In this paper we deal with *global scheduling*[1] of arbitrary deadline systems upon *identical parallel machines*, i.e., all the processors are identical in the sense that they have the same computing power.

Related research. The problem of scheduling periodic constrained deadline task systems was originally proposed in [3]. A better understanding of this problem was provided through [4, 5, 6]. All these papers considered results on

*Post-doctorante du F.N.R.S.

[1] task and job migration are allowed.

978-3-9810801-2-4/DATE07 © 2007 EDAA

feasibility tests or improved algorithms in order to increase processors utilization for constrained deadline systems. For the case of scheduling periodic tasks with arbitrary deadlines, Goossens et al. [7] showed that many uniprocessor results do not remain for multiprocessor scheduling. For instance the synchronous case is not necessarily the worst case for identical processors. Moreover, the feasibility interval of the uniprocessor case given in [8] does not stand for identical processors. For that reason we shall present and prove correct in this paper feasibility intervals and related properties for periodic tasks with arbitrary deadlines on identical multiprocessors.

This research. In this paper, we adapt the results concerning feasibility intervals for periodic tasks with constrained deadlines on uniform processors [6] to periodic tasks with arbitrary deadlines on identical processors. We show that any feasible schedules of arbitrary deadline periodic task systems on identical processors repeat from some point. For synchronous arbitrary deadline periodic task systems on identical processors using fixed-priority scheduling[2] we show that the schedule repeats from the origin. Then we give a feasibility interval for the case of global fixed-priority scheduling of these systems. Then we prove a more precise and useful result, the main contribution of this paper: any feasible schedules of fixed-priority scheduling of asynchronous arbitrary deadline periodic task systems on identical parallel processors are periodic from a specific point (or possibly before). We also characterize that point and we provide a feasibility interval for those systems.

Organization. This paper is organized as follows. In Section 2, we introduce our model of computation. In Section 3, we show that any feasible schedules of periodic task systems are periodic from some point. In Section 3.1, we consider the specific case of synchronous periodic task systems. In Section 4, we present our main result: a feasibility interval for asynchronous periodic task sets using global fixed-priority scheduling. We conclude in Section 5.

2 Definitions and assumptions

We consider the scheduling of periodic task systems. A system τ is composed by n periodic tasks $\tau_1, \tau_2, \ldots, \tau_n$, each task is characterized by a period T_i, a relative deadline D_i, a worst-case execution time C_i and an offset O_i. The costs relatively to migrations or context switches are neglected in the sense that the worst-case execution times of tasks contain them. Such a periodic task generates an infinite sequence of jobs, with the k^{th} job arriving at time-instant $O_i + (k-1)T_i$ $(k = 1, 2, \ldots)$, having a worst-case execution requirement of C_i units, and a hard deadline at time instant $O_i + (k-1)T_i + D_i$.

[2]the priorities are assigned to the tasks beforehand, at run-time each request inherits of its task priority and remains constant.

In some cases, we shall consider the more general problem of scheduling set of jobs, each job $J_j = (r_j, e_j, d_j)$ is characterized by a release time r_j, an execution time e_j and an absolute deadline d_j. The job J_j must execute for e_j time units over the interval $[r_j, d_j)$. A job is active from its release time to its completion time.

We denote by $\tau^{(i)} \stackrel{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_i\}$, by $O_{\max} \stackrel{\text{def}}{=} \max\{O_1, O_2, \ldots, O_n\}$, by $P_i \stackrel{\text{def}}{=} \text{lcm}\{T_1, T_2, \ldots, T_i\}$ and $P \stackrel{\text{def}}{=} P_n$.

A system τ is said to be *feasible* upon a multiprocessor platform if there exists at least one schedule in which all tasks meet their deadlines. If A is an algorithm which schedules τ upon a multiprocessor platform to meet its deadlines, then the system τ is said to be A-feasible.

We consider in this paper m identical processors $\{p_1, p_2, \ldots, p_m\}$.

The time model is a discrete model, i.e. the characteristics of the tasks and the time are integers.

We define now the notions of the state of the system and the schedule.

Definition 1 (State of the system $\theta(t)$) *For any arbitrary deadline system $\tau = \{\tau_1, \ldots, \tau_n\}$ we define the* state $\theta(t)$ *of the system τ at instant t as $\theta : \mathbb{N} \to (\{-1, 0, 1\} \times \mathbb{N}^2)^n$ with $\theta(t) \stackrel{\text{def}}{=} (\theta_1(t), \theta_2(t), \ldots, \theta_n(t))$ where*

$$\theta_i(t) \stackrel{\text{def}}{=} \begin{cases} (-1, t_1, 0), & \text{if no job of task } \tau_i \text{ was activated before or at } t \text{ and it remains } t_1 \text{ time units until the first activation of } \tau_i. \text{ (We have } 0 < t_1 \leq O_i.\text{);} \\ (n_1, t_2, t_3), & \text{where } t_2 \text{ is the time elapsed at instant } t \text{ since the last action of the oldest active job of } \tau_i. \text{ If there are } n_1 \neq 0 \text{ active jobs of } \tau_i \text{ then } t_3 \text{ units were already executed from the oldest active job. If there is no active job of } \tau_i \text{ at } t \text{ but at least one job of } \tau_i \text{ was already actived before } t, \text{ then } n_1 = t_3 = 0. \text{ (We have } 0 \leq n_1 \leq \lceil \frac{D_i}{T_i} \rceil, 0 \leq t_2 < T_i \text{ and } 0 \leq t_3 < C_i.\text{)} \end{cases}$$

Notice that at any instant t several jobs of the same task might be active and we consider that the oldest job is scheduled first, i.e. the FIFO rule is used to serve the various jobs of given task.

Definition 2 (Schedule $\sigma(t)$) *For any task system $\tau = \{\tau_1, \ldots, \tau_n\}$ and any set of m processors $\{p_1, \ldots, p_m\}$ we define the* schedule $\sigma(t)$ *of system τ at instant t as $\sigma : \mathbb{N} \to$*

$\{0, 1, \ldots, n\}^m$ *where* $\sigma(t) \stackrel{\text{def}}{=} (\sigma_1(t), \sigma_2(t), \ldots, \sigma_m(t))$ *with*

$$\sigma_j(t) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if there is no task scheduled on } p_j \\ & \text{at instant } t; \\ i, & \text{if } \tau_i \text{ is scheduled on } p_j \text{ at instant } t \end{cases}$$

$\forall 1 \leq j \leq m$.

Notice that Definition 2 can be extended trivially to the scheduling of a set of jobs.

In this work, we consider that task parallelism is forbidden: a task cannot be scheduled at the same instant on different processors, i.e. $\nexists j_1 \neq j_2 \in \{1, 2, \ldots, m\}$ and $t \in \mathbb{N}$ such that $\sigma_{j_1}(t) = \sigma_{j_2}(t) \neq 0$.

The scheduling algorithms considered in this paper are *deterministic* with the following definition:

Definition 3 (Deterministic algorithms) *A scheduling algorithm is said to be* deterministic *if it generates a unique schedule for any given set of jobs .*

Moreover, we shall assume that the decision of the scheduling algorithm at time t is not based on the past, nor on the actual time t, but only on the characteristics of active jobs and on the state of the system at time t. More formally, we consider *memoryless* schedulers.

Definition 4 (Memoryless algorithms) *A scheduling algorithm is said to be* memoryless *if the scheduling decision made by it at time t depends only on the characteristics of active tasks and on the current state of the system, i.e., on $\theta(t)$.*

Consequently, for memoryless and deterministic schedulers we have the following property:

$$\forall t_1, t_2 \text{ such that } \theta(t_1) = \theta(t_2) \text{ then } \sigma(t_1) = \sigma(t_2).$$

We formalize the notion of *work-conserving* algorithms and the notion of *feasibility* interval.

Definition 5 (Work-conserving algorithm) *A* work-conserving algorithm *is a scheduling algorithm that never idles a processor while there is at least one active task.*

Definition 6 (Feasibility interval) *For any task system $\tau = \{\tau_1, \ldots, \tau_n\}$ and any set of m processors $\{p_1, \ldots, p_m\}$, the* feasibility interval *is a finite interval such that if no deadline is missed while considering only requests within this interval then no deadline will ever be missed.*

The fact that work-conserving and priority-driven algorithms are *predictable* on identical platforms allows us to give a feasibility interval even if the actual execution times are less than C_i (which is only an *upper* bound of the actual execution time of the task requests). But first we need some additional definitions in order to present the predictability result given in [6].

Definition 7 (Priority-driven algorithms) *A scheduling algorithm is a* priority-driven *algorithm if and only if it satisfies the condition that for every pair of jobs J_i and J_j, if J_i has higher priority than J_j at some time instant, then J_i always has higher priority than J_j.*

Definition 8 (Predictable algorithms) *Let $J = \{J_1, \ldots, J_\ell\}$ a set of jobs with $J_i = (r_i, e_i, d_i)$. Let $S(J_i)$ be the time instant at which J_i begins its execution in a schedule and, similarly, let $F(J_i)$ be the time instant at which J_i completes its execution in a schedule. Let $J' = \{J'_1, \ldots, J'_\ell\}$ be a set of jobs obtained from J such that $J'_i = (r_i, e'_i, d_i)$ and $e'_i \leq e_i, \forall i$. A scheduling algorithm A is* predictable *if and only if we have that $S(J'_i) \leq S(J_i)$ and $F(J'_i) \leq F(J_i)$, $\forall i$ $(1 \leq i \leq \ell), \forall J$ and $\forall J'$ (as defined above).*

Theorem 1 ([6]) *Work-conserving and priority-driven algorithms are predictable on identical platforms.*

3 Periodicity of deterministic and memoryless scheduling algorithms

In this section, we shall show that feasible schedules of arbitrary deadline task systems obtained using deterministic and memoryless algorithm are periodic from some point (Theorem 2), assuming that the execution times of tasks are constant.

Theorem 2 *For any deterministic and memoryless algorithm A, if an asynchronous arbitrary deadline system τ is A-feasible on m processors, then the A-feasible schedule of τ on m processors is finally periodic, i.e. from some point the schedule repeats. (Assuming that the execution time of each task is constant.)*

Proof. First notice that from $t_0 \geq O_{\max}$ all tasks are released, and the configuration $\theta_i(t)$ of each task is a triple of finite integers (α, β, γ) with $0 \leq \alpha \leq \lceil \frac{D_i}{T_i} \rceil$, $0 \leq \beta < \max_{1 \leq i \leq n} T_i$ and $0 \leq \gamma < \max_{1 \leq i \leq n} C_i$. Therefore there is a finite number of different system states, hence we can find two instants t_1 and t_2 ($t_2 > t_1 \geq t_0$) with the same state of the system. The schedule repeats from that instant with a period dividing $t_2 - t_1$, since the scheduler is deterministic and memoryless. □

3.1 The particular case of synchronous periodic systems

In this section we deal with the particular case of synchronous arbitrary deadlines task systems and we show the periodicity of feasible schedules obtained using preemptive fixed-priority scheduling algorithms.

In the following without loss of generality we consider the tasks ordered in decreasing order of their priorities $\tau_1 > \tau_2 > \cdots > \tau_n$.

Lemma 3 *For any preemptive fixed-priority algorithm A and any synchronous arbitrary deadline system τ on m processors, if no deadline is missed in the time interval $[0, P)$ and the system is in the same state at time instants 0 and P, then the schedule of τ is periodic with a period P that begins at instant 0. (Assuming that the execution time of each task is constant.)*

Proof. Since at time instants 0 and P the system is in the same state, i.e. $\theta(0) = \theta(P)$, then at time instants 0 and P a preemptive fixed-priority algorithm will make the same scheduling decision and the scheduled repeats from 0 with a period equal to P. Notice also that we have proved that the system is A-feasible. □

Theorem 4 *For any preemptive fixed-priority algorithm A and a synchronous arbitrary deadline system τ on m processors, if there is more than one active job of the same task at P, then τ is not A-feasible.*

Proof. Since there is more than one active job of the same task at P, then we may consider $i_0 \in \{1, 2, \ldots, n\}$ to be the smallest task index such that τ_{i_0} has at least two active jobs at P_{i_0}. In order to prove the lemma we will prove that τ_{i_0} will miss a deadline.

Since $\theta(0) = \theta(P_{i_0-1})$ and by definition of i_0 from Lemma 3 we have that the time instants, such that at least one processor is available, are periodic with a period P_{i_0-1}, i.e., the schedule $\sigma^{(i_0-1)}$ obtained by considering only the task subset $\tau^{(i_0-1)}$ is periodic with a period P_{i_0-1}. Moreover since P_{i_0} is a multiple of P_{i_0-1}, then the schedule $\sigma^{(i_0-1)}$ obtained by considering only the task subset $\tau^{(i_0-1)}$ is periodic with a period P_{i_0}. Therefore in each time interval $[k \cdot P_{i_0}, (k+1)P_{i_0})$ with $k \geq 0$ after scheduling $\tau_1, \tau_2, \ldots, \tau_{i_0-1}$ there is the same number t_{i_0} of time instants such that at least one processor is available and where τ_{i_0} is scheduled. At time instant P_{i_0}, since the task parallelism is forbidden, there are $\frac{P_{i_0}}{T_{i_0}} C_{i_0} - t_{i_0}$ remaining units for execution of τ_{i_0} and, consequently, at each time instant $(k+1) \cdot P_{i_0}$ there will be $k \cdot (\frac{P_{i_0}}{T_{i_0}} C_{i_0} - t_{i_0})$ remaining units for execution of τ_{i_0}. Consequently we can find $k_{i_0} = \lceil \frac{D_{i_0}}{\frac{P_{i_0}}{T_{i_0}} C_{i_0} - t_{i_0}} \rceil$ such that the job actived at $(k_{i_0}+1)P_{i_0}$ will miss its deadline since it cannot be scheduled before older jobs of τ_{i_0} and there are $k_{i_0}(\frac{P_{i_0}}{T_{i_0}} C_{i_0} - t_{i_0}) \geq D_{i_0}$ remaining units for execution of τ_{i_0} at $(k_{i_0}+1)P_{i_0}$.

Since we consider fixed-priority scheduling, then the tasks τ_i with $i > i_0$ will not interfere with the higher priority tasks already scheduled, particularly with τ_{i_0} that misses its deadline, and consequently the system is not A-feasible. □

Corollary 5 *For any preemptive fixed-priority algorithm A and any synchronous arbitrary deadline system τ, τ is A-feasible on m processors if and only if: (1) all deadlines are met in the interval $[0, P)$, and (2) $\theta(0) = \theta(P)$.*

Proof. The result is a direct consequence of Theorem 4 and Theorem 1, since fixed-priority schedulers are priority-driven. □

Similarly to the uniprocessor case (see [9], p. 57 for details) both conditions of Corollary 5 are necessary for the feasibility of a system. The following example shows that a system with no deadline missed in $[0, P)$ can be infeasible. It also shows that the natural generalization of the uniprocessor property: $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \Rightarrow \theta(0) = \theta(P)$ is not valid in the multiprocessor case since $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq m \not\Rightarrow \theta(0) = \theta(P)$.

We consider the synchronous arbitrary deadline task system $\tau = \{\tau_1 > \cdots > \tau_4\}$ with $\tau_1 = (3, 5, 5)$, $\tau_2 = (1, 5, 5)$, $\tau_3 = (2, 5, 7)$ and $\tau_4 = (2, 4, 10)$, where $\tau_i = (C_i, T_i, D_i)$. The schedule on 2 processors is given in Figure 1 and the 7^{th} job of τ_4 misses its deadline at time instant 34. The utilization of processors is $\frac{34}{20} < 2$ and no deadline is missed in the time interval $[0, 20)$, while the system is infeasible. In Figure 1 ⌐ indicates the time instant when a job is actived and its associate deadline. □ indicates that the time unit of a job is scheduled on the first processor and ▨ on the second processor.

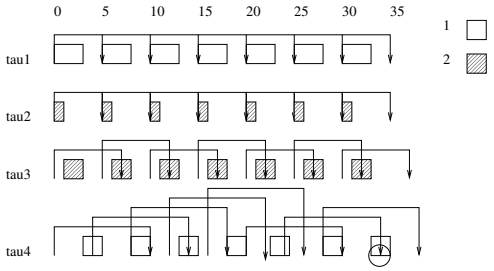

Figure 1. Fixed-priority schedule of task system given as example

4 Fixed-priority scheduling of asynchronous systems

In this section we present our main result: any feasible schedule on m identical processors of asynchronous arbitrary deadline systems, obtained using preemptive fixed-priority algorithms, is periodic from some point (Theorem 8), assuming that the execution time of each task τ_i is constant. Moreover, we define a feasibility interval for this case (Corollary 9).

We introduce now the availability of the processors for any schedule $\sigma(t)$.

Definition 9 (Availability of the processors $a(t)$) *For any task system $\tau = \{\tau_1, \ldots, \tau_n\}$ and any set of m processors $\{p_1, \ldots, p_m\}$ we define the* availability of the processors *$a(t)$ of system τ at instant t as the set of available processors $a(t) \stackrel{\text{def}}{=} \{j \mid \sigma_j(t) = 0\} \subseteq \{1, \ldots, m\}$.*

We denote by δ_i^k the k^{th} job of task τ_i which becomes active at time instant $R_i^k \stackrel{\text{def}}{=} O_i + (k-1)T_i$.

Definition 10 *For any task τ_i, we define $\epsilon_i^k(t)$ the amount of time already executed from the k^{th} job of task τ_i in the interval $[R_i^k, t)$.*

The following lemma and corollary extend results given for arbitrary deadline task systems in the *uniprocessor* case (see [9], p. 55 for details).

Lemma 6 *For any preemptive fixed-priority algorithm A and an asynchronous arbitrary deadline system τ on m processors, we have that: for each task τ_i, for any time instant $t \geq O_i$ and k such that $R_i^k \leq t \leq R_i^k + D_i$, if there is no deadline missed up to time $t+P$, then $\epsilon_i^k(t) \geq \epsilon_i^{k+h_i}(t+P)$ with $h_i \stackrel{\text{def}}{=} \frac{P}{T_i}$.*

Proof. The proof is made by contradiction. Notice first that the function ϵ_i^k is a non-decreasing discrete step function with $0 \leq \epsilon_i^k(t) \leq C_i, \forall t$ and $\epsilon_i(R_i^k) = 0 = \epsilon_i(R_i^{k+h_i})$.

We assume that a first time instant t exists such that there are j and k with $R_j^k \leq t \leq R_j^k + D_j$ and $\epsilon_j^k(t) < \epsilon_j^{k+h_j}(t+P)$. This assumption implies that there is a time instant t' with $R_j^k \leq t' < t$ such that $\delta_j^{k+h_j}$ is scheduled at $t'+P$ while δ_j^k is not scheduled at t'. We obtain that either m jobs of m higher priority tasks are scheduled at t', or an older job of τ_j is scheduled at t'. Among these jobs there is one job $\delta_{i_\ell}^{k'_\ell}$ with $i_\ell \in \{1, 2, \ldots, j\}$ that is not scheduled at $t'+P$. This implies that $\epsilon_{i_\ell}^{k'_\ell}(t') < \epsilon_{i_\ell}^{k'_\ell + \frac{P}{T_{i_\ell}}}(t'+P) = C_{i_\ell}$ which is in contradiction with the fact that t is the first such time instant. □

Corollary 7 *For any preemptive fixed-priority algorithm A and an asynchronous arbitrary deadline system τ on m processors, we have that: for each task τ_i, for any time instant $t \geq O_i$, if there is no deadline missed up to time $t+P$, then either $(\alpha_i(t) < \alpha_i(t+P))$ or $[\alpha_i(t) = \alpha_i(t+P)$ and $\gamma_i(t) \geq \gamma_i(t+P)]$, where by the triple $(\alpha_i(t), \beta_i(t), \gamma_i(t))$ we denoted $\theta_i(t)$.*

Proof. If $\alpha_i(t) = 0$, then either $\alpha_i(t+P) > 0$ or $\alpha_i(t+P) = 0 = \beta_i(t+P) = \beta_i(t)$. Otherwise, $\alpha_i(t) = n_i(t) - m_i(t)$ where $n_i(t)$ is the number of jobs actived before or at t, and $m_i(t)$ is the number of jobs that have completed their execution before or at t. We have $n_i(t+P) = n_i(t) + \frac{P}{T_i}$ and from Lemma 6 we obtain that $m_i(t+P) \leq m_i(t) + \frac{P}{T_i}$. Consequently $\alpha_i(t+P) \geq \alpha_i(t)$, and if $\alpha_i(t) = \alpha_i(t+P)$ then $m_i(t+P) = m_i(t) + \frac{P}{T_i}$, and $\beta_i(t) = \epsilon^{m_i(t)+1} \geq \epsilon_i^{m_i(t)+1+\frac{P}{T_i}}(t+P) = \beta_i(t+P)$. □

Theorem 8 *For any preemptive fixed-priority algorithm A, if an asynchronous arbitrary deadline system τ is A-feasible on m processors, then the A-feasible schedule of τ is periodic with a period P from instant t_n where t_n are defined inductively as follows:*

- $t_1 \stackrel{\text{def}}{=} O_1$
- $t_i \stackrel{\text{def}}{=} \max\left\{O_i, O_i + \lceil \frac{t_{i-1} - O_i}{T_i} \rceil T_i\right\} + P_i, \qquad (i > 1)$

(Assuming that the execution time of each task is constant.)

Proof. The proof is made by induction by n (the number of tasks). We denote by $\sigma^{(i)}$ the schedule obtained by considering only the task subset $\tau^{(i)}$, the first higher priority i tasks $\{\tau_1, \ldots, \tau_i\}$, and by $a^{(i)}$ the corresponding availability of the processors. Our inductive hypothesis is the following: the schedule $\sigma^{(k)}$ is periodic from t_k with a period P_k, for all $1 \leq k \leq i$.

The property is true in the base case: $\sigma^{(1)}$ is periodic from $t_1 = O_1$ with period P_1, for $\tau^{(1)} = \{\tau_1\}$: since we consider feasible systems, at instant $P_1 + O_1 = T_1 + O_1$ the previous job of τ_1 has finished its execution ($C_1 \leq T_1$) and the schedule repeats.

We shall now show that any A-feasible schedule of $\tau^{(i+1)}$ is periodic with period P_{i+1} from t_{i+1}.

Since $\sigma^{(i)}$ is periodic with a period P_i from t_i the following equation is verified:

$$\sigma^{(i)}(t) = \sigma^{(i)}(t + P_i), \forall t \geq t_i. \tag{1}$$

We denote by $t_{i+1} \stackrel{\text{def}}{=} \max\{O_{i+1}, O_{i+1} + \lceil \frac{t_i - O_{i+1}}{T_{i+1}} \rceil T_{i+1}\} + P_{i+1}$ the time instant obtained by adding P_{i+1} to the time instant which corresponds to the first activation of τ_{i+1} after t_i.

Since the tasks in $\tau^{(i)}$ have higher priority than τ_{i+1}, then the scheduling of τ_{i+1} will not interfere with higher priority tasks which are already scheduled. Therefore, we may build $\sigma^{(i+1)}$ from $\sigma^{(i)}$ such that the tasks $\tau_1, \tau_2, \ldots, \tau_i$ are scheduled at the very same instants and on the very same processors as there were in $\sigma^{(i)}$. We apply now the induction step: for all $t \geq t_i$ in $\sigma^{(i)}$ we have $a^{(i)}(t) = a^{(i)}(t+P_i)$ the availability of the processors repeats. Notice that at the instants t and $t + P_i$ the available processors (if any) are the same. Hence at only these instants task τ_{i+1} *may* be executed in the time interval $[t_{i+1}, t_{i+1} + P_{i+1})$.

The instants t such that $t_{i+1} \leq t < t_{i+1} + P_{i+1}$, where τ_{i+1} may be executed in $\sigma^{(i+1)}$, are periodic with period P_{i+1}, since P_{i+1} is a multiple of P_i and $t_{i+1} \geq t_i$. We prove now by contradiction that the system is in the same state at time instant t_{i+1} and $t_{i+1} + P_{i+1}$. We suppose that $\theta(t_{i+1}) \neq \theta(t_{i+1} + P_{i+1})$.

We first prove that $\nexists t \in [t_{i+1}, t_{i+1} + P_{i+1})$ such that at t there is at least one available processor in $\sigma^{(i)}$ and no job of τ_{i+1} is scheduled at t in $\sigma^{(i+1)}$. If there is such an instant t', then from Corollary 7 we have that $\theta(t' - P_{i+1}) = \theta(t')$ since from the inductive hypothesis (notice that P_{i+1} is multiple of P_i) and since $t' - P_{i+1} \geq t_{i+1} - P_{i+1} \geq t_i \geq \cdots \geq t_1$ we obtain that $\theta_k(t' - P_{i+1}) = \theta_k(t')$ for $1 \leq k \leq i$. Consequently, $\theta(t_{i+1}) = \theta(t_{i+1} + P_{i+1})$ which is in contradiction with our assumption.

Secondly, since $\theta_{i+1}(t_{i+1}) \neq \theta_{i+1}(t_{i+1} + P_{i+1})$ then from Corollary 7 we have that either there are less active jobs at t_{i+1} than at $t_{i+1} + P_{i+1}$, or if there is the same number of active jobs of t_{i+1} then the oldest active job at t_{i+1} was executed for more time units than the oldest active at $t_{i+1} + P_{i+1}$. Therefore since $\nexists t \in [t_{i+1}, t_{i+1} + P_{i+1})$ such that at t there is at least one processor available in $\sigma^{(i)}$ and no job of τ_{i+1} is scheduled at t in $\sigma^{(i+1)}$, then we have that there are no sufficient time instants when at least one processor is available to schedule all the jobs actived of τ_{i+1} in the time interval $[t_{i+1}, t_{i+1} + P_{i+1})$. We obtain that the system is not feasible, which is in contradiction with our assumption of τ being feasible.

Consequently $\theta(t_{i+1}) = \theta(t_{i+1} + P_{i+1})$, moreover by definition of t_{i+1} (which corresponds to an activation of τ_{i+1}) the task activations repeat from t_{i+1} which proves the property. □

Now we have the material to define a feasibility interval for asynchronous arbitrary deadline periodic systems.

Corollary 9 *For any preemptive work-conserving fixed-priority algorithm A and for any A-feasible asynchronous arbitrary deadline system τ on m processors, $[0, t_n + P)$ is a feasibility interval where t_i are defined inductively in Theorem 8.*

Proof. The Corollary 9 is a direct consequence of Theorem 8 and Theorem 1, since fixed-priority algorithms are priority-driven. □

Notice that the length of our feasibility interval is proportional to P (the least common multiple of the periods) which is unfortunately also the case of most feasibility intervals for the *simpler uni*processor scheduling problem (and for identical or simpler task models). In practice, the periods are usually *harmonics* which limits fairly the term P.

5 Conclusion and future work

In this work we adapted some results concerning the periodicity of feasible schedules and the feasibility intervals from the uniprocessor case to the multiprocessor case for arbitrary deadline task systems. We showed that any feasible schedule of a periodic task system on identical parallel processors repeats from some point. We provided feasibility intervals for synchronous arbitrary deadline task systems using preemptive fixed-priority algorithms. Then, we presented the main contribution of this paper: feasibility intervals for asynchronous arbitrary deadline task systems obtained using preemptive fixed-priority algorithms.

As future work we are interested in extending the results to heterogeneous multiprocessor platforms. We are also interested in obtaining results concerning feasibility intervals for asynchronous arbitrary deadline task systems obtained using preemptive fixed-priority algorithms on multiprocessor platforms when active jobs of the same task can be scheduled at the same time on *different* processors.

References

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

[2] W.K. Shih, J.W.S. Liu, and C.L. Liu. Scheduling periodic jobs with deffered deadlines. Technical report, University of Illinois, 1990.

[3] C.L. Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary 37-60(II)*, pages 28–31, 1969.

[4] B. Andersson, S. Baruah, and J. Jansson. Static-priority scheduling on multiprocessors. *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 193–202, 2001.

[5] J. Anderson, P. Holman, and A. Srinivasan. Fair scheduling of real-time tasks on multiprocessors. *Handbook of Scheduling*, 2005.

[6] L. Cucu and J. Goossens. Feasibility intervals for fixed-priority real-time scheduling on uniform multiprocessors. *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 397–405, 2006.

[7] J. Goossens, S. Funk, and S. Baruah. EDF scheduling on multiprocessors: some (perhaps) counterintuitive observations. *Proceedings of the 8th International Conference on Real-Time Computing Systems and Applications*, pages 321–330, 2002.

[8] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *IEEE Real-Time Systems Symposium*, pages 201–213, 1990.

[9] J. Goossens. *Scheduling of Hard Real-Time Periodic Systems with Various Kinds of Deadline and Offset Constraints*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 1999.