Utilization Bounds for EDF Scheduling on Real-Time Multiprocessor Systems

J.M. López, J.L. Díaz and D.F. García
Departamento de Informática, Universidad de Oviedo, Gijón 33204, Spain

Abstract. The utilization bound for Earliest Deadline First scheduling is extended from uniprocessors to homogeneous multiprocessor systems with partitioning strategies. First results are provided for a basic task model, which includes periodic and independent tasks with deadlines equal to periods. Since the multiprocessor utilization bounds depend on the allocation algorithm, different allocation algorithms have been considered, ranging from simple heuristics to optimal allocation algorithms.

As multiprocessor utilization bounds for EDF scheduling depend strongly on task sizes, all these bounds have been obtained as a function of a parameter which takes task sizes into account.

Theoretically, the utilization bounds for multiprocessor EDF scheduling can be considered a partial solution to the bin-packing problem, which is known to be NP-complete.

The basic task model is extended to include resource sharing, release jitter, deadlines less than periods, aperiodic tasks, non-preemptive sections, context switches and mode changes.

Keywords: multiprocessor scheduling, partitioning, bin-packing problem, Earliest Deadline First scheduling, multiprocessor utilization bounds

1. Introduction

Real-time systems theory supplies many results about uniprocessors but few about multiprocessors. One of the outstanding results about uniprocessors is the optimality of Earliest Deadline First (EDF) scheduling for any kind of tasks (Dertouzos, 1974). Unfortunately, EDF scheduling is not optimal on multiprocessor systems (Dertouzos and Mok, 1989).

A new issue arises on multiprocessor scheduling; that is which processor executes each task at a given time. There are two major strategies to deal with this problem: *partitioning strategies*, and *global strategies* (Oh and Son, 1995). In a partitioning strategy, once a task is allocated to a processor, it is executed exclusively on that processor. In a global strategy, any instance of a task can be executed on any processor, or even be preempted and moved to a different processor before it is completed.

Theoretically, global strategies provide in general higher schedulability than partitioning strategies. However, partitioning strategies have


© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

several advantages over global strategies. Firstly, the scheduling overhead associated with a partitioning strategy is lower than the overhead associated with a global strategy. Secondly, partitioning strategies allow us to apply well-known uniprocessor scheduling algorithms to each processor. Furthermore, Earliest Deadline First (EDF) scheduling, which is an optimal uniprocessor scheduling algorithm, perform poorly when extended to global multiprocessor scheduling. The utilization bound associated with global EDF multiprocessor scheduling is not higher than one for any number of processors (Dall and Liu, 1978).

In this paper we follow the partitioning strategy, and assume that all the tasks allocated to a processor are preemptively scheduled using EDF, as this is the optimal scheduling algorithm for uniprocessors. Once the scheduler has been chosen, the only degree of freedom is the allocation algorithm.

The problem of allocating a set of tasks to a set of processors is analogous to the bin-packing problem. In this case, the tasks are the objects to pack, of size equal to their utilization factors. The bins are processors with a capacity of one for EDF scheduling (Liu and Layland, 1973). The bin-packing problem is known to be NP-hard in the strong sense (Garey and Johnson, 1979). Thus, searching for optimal allocation algorithms is not practical. Several allocation algorithms have been proposed (Oh and Son, 1995; Garey and Johnson, 1979; Burchard et al., 1995; Dall and Liu, 1978; Sáez et al., 1998).

Two different approaches are followed in the literature to establish the schedulability associated with a given allocation algorithm: simulation approaches and theoretical approaches.

In the simulation approach, task sets are randomly generated. Next, the average number of processors required to allocate task sets of given total utilization is obtained. Uniprocessor exact tests (Sáez et al., 1998), or uniprocessor sufficient tests (Lauzac et al., 1998) are commonly used to decide whether a given group of tasks fits into one processor. Nevertheless, simulation results should be considered carefully, since randomly generated task sets may not be representative of those that appear in practice.

The traditional theoretical approach focuses on the calculation of the metric (N_{AA}/N_{OPT}), for pairs *uniprocessor scheduling algorithm-allocation algorithm* (Oh and Son, 1995; Dall and Liu, 1978; Garey and Johnson, 1979; Burchard et al., 1995; Davari and Dhall, 1986a; Davari and Dhall, 1986b). This metric gives the relationship between the number of processors required to schedule a task set using a given allocation algorithm, AA, and the number of processors required using the optimal allocation algorithm,OPT. This metric is useful in order to compare

different allocation algorithms, but not to perform a schedulability test (López et al., 2003).

A new theoretical approach consists of calculating the utilization bounds associated with *scheduling algorithm-allocation algorithm* pairs, analogous to those known for uniprocessors. This approach has several interesting features:

- Firstly, it allows us to carry out fast schedulability tests.
- Secondly, it allows us to quantify the influence of certain parameters, such as the number of processors, on schedulability.

The major disadvantage of this approach is the sufficient but not necessary character of the associated schedulability tests. This approach was followed by Oh and Baker (1998) and López et al. (2003) for multiprocessor RM scheduling.

Our work provides utilization bounds for multiprocessor EDF scheduling and deals with:

- Complex task models, including periodic and aperiodic tasks, release jitter, arbitrary deadlines, resource sharing and mode changes.
- Arbitrary allocation algorithms, ranging from optimal allocation algorithms to simple heuristics.
- Task sizes: by taking into account task sizes, the utilization bounds can be greatly incremented.

The rest of the paper is organized as follows. Section 2 defines the basic system model dealt with. Section 3 provides limits on the utilization bounds for arbitrary allocation algorithms under the basic system model. The utilization bounds for the basic system model and different allocation algorithms are provided in Section 4. Section 5 analyzes the expressions of the utilization bounds. Section 6 extends the basic system model, generalizing the task model, and calculates new utilization bounds for the extensions. Finally, Section 7 presents our conclusions and future work.

2. Basic system model

The task set consists of m independent and periodic tasks $\{\tau_1, \ldots, \tau_m\}$, of computation times $\{C_1, \ldots, C_m\}$, periods $\{T_1, \ldots, T_m\}$, and hard deadlines equal to the task periods. The utilization factor u_i of any task τ_i, defined as $u_i = C_i/T_i$, is assumed to be $0 < u_i \leq \alpha \leq 1$, where

α is the maximum reachable utilization factor among all tasks. Thus, α is a parameter of the task set which takes the "task sizes" into account. The total utilization of the task set, denoted by U, is the sum of the utilization factors of the tasks that compose it.

Tasks are allocated to an array of n identical processors $\{P_1, \ldots, P_n\}$, and are executed independently of each other. Once a task is allocated to a processor, it is executed only on that processor. Within each processor, tasks are scheduled preemptively using EDF. The allocation is carried out using Reasonable Allocation algorithms (RA). A reasonable allocation algorithm is one which fails to allocate a task only when there is no processor in the system which can hold the task.

The uniprocessor utilization bound for EDF scheduling of periodic and independent tasks of periods equal to deadlines is 1.0 (Liu and Layland, 1973). This means that any subset of the tasks is schedulable on one processor if and only if $U \leq 1$. Thus, a task of utilization factor u_i fits into processor P_j, which already has m_j tasks allocated to it with total utilization U_j, if the $(m_j + 1)$ tasks are schedulable, i.e, if $1 - U_j \geq u_i$.

Using the schedulability condition $U \leq 1$, a reasonable allocation algorithm is one which fails to allocate a task of utilization factor u_i to a multiprocessor made up of n processors, only when the task does not fit into any processor, i.e,

$$1 - U_j < u_i \qquad \text{for } j = 1, \ldots, n \tag{1}$$

Examples of reasonable allocation algorithms are First Fit (FF), Best Fit (BF) and the optimal allocation algorithm (OPT).

Within the Reasonable Allocation (RA) algorithms we define a class, termed Reasonable Allocation Decreasing (RAD), made up of all the reasonable allocation algorithms fulfilling the following conditions:

- Tasks are ordered by decreasing utilization factors before making the allocation, i.e, $u_1 \geq u_2 \geq \cdots \geq u_m$.
- Tasks are allocated sequentially, that is, task τ_1 is allocated first, next task τ_2, and so on until task τ_m.

3. Limits on the utilization bounds

The multiprocessor utilization bound associated with any reasonable allocation algorithm, RA, and multiprocessor EDF scheduling is denoted by $U_{\text{wc}}^{\text{EDF-RA}}$. Any task set of total utilization $U \leq U_{\text{wc}}^{\text{EDF-RA}}$ is schedulable using RA allocation and EDF scheduling on all processors,

while a task set of total utilization $U > U_{\text{wc}}^{\text{EDF-RA}}$ may or may not be schedulable. The multiprocessor utilization bound $U_{\text{wc}}^{\text{EDF-RA}}$ depends on the allocation algorithm, RA. Therefore, $U_{\text{wc}}^{\text{EDF-RA}}$ is in the interval $[L_{\text{EDF}}, H_{\text{EDF}}]$, defined as follows:

$$L_{\text{EDF}} = \min_{RA} U_{\text{wc}}^{\text{EDF-RA}}$$
$$H_{\text{EDF}} = \max_{RA} U_{\text{wc}}^{\text{EDF-RA}}$$

The calculation of this interval gives the worst and best utilization bounds we can expect from all the reasonable allocation algorithms.

Before calculating the expressions of L_{EDF} and H_{EDF}, it is necessary to introduce the parameter β. β is the maximum number of tasks of utilization factor α which fit into one processor under EDF scheduling. β can be expressed as a function of α

$$\beta = \lfloor 1/\alpha \rfloor$$

Any multiprocessor made up of n processors can allocate at least βn tasks of arbitrary utilization factors (less than or equal to α). Thus, any task set fulfilling $m \leq \beta n$ is trivially schedulable using EDF scheduling together with any reasonable allocation algorithm.

Figure 1 depicts β as a function of α, showing also the sufficient schedulability condition $m \leq \beta n$. For example, if α is in the interval $(1/3, 1/2]$ then $\beta = 2$. In this case, the task set is schedulable if it has $2n$ tasks or less.

Henceforth, we will assume $m > \beta n$, as otherwise there would be no point in obtaining the utilization bounds.

Theorem 1 will provide a lower limit on the utilization bound associated with any reasonable allocation algorithm and multiprocessor EDF scheduling. Section 4.1 will present the utilization bound for one reasonable allocation algorithm, the Worst Fit (WF) heuristic, which coincides with the previous lower limit. Therefore, L_{EDF} and the utilization bound for WF allocation coincide.

Theorem 2 will provide an upper limit on the utilization bound associated with any allocation algorithm, reasonable or not, and multiprocessor EDF scheduling. Section 4.2 will present the utilization bound associated with some reasonable allocation algorithms, the heuristics in the class RAD, which coincides with the previous upper limit. Therefore, H_{EDF} and the utilization bound for the class RAD coincide. Furthermore, since the upper limit given by Theorem 2 applies to any allocation algorithm, reasonable or not, H_{EDF} is the maximum utilization bound among all the allocation algorithms.

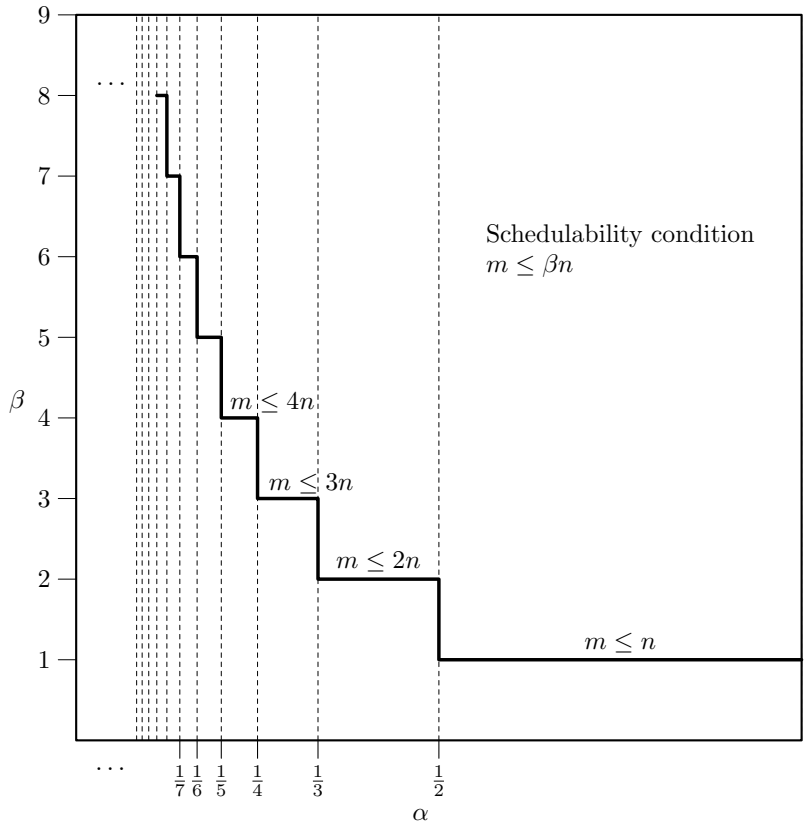

Figure 1. Representation of the function $\beta(\alpha)$, and the associated schedulability condition.

THEOREM 1. *Let RA be any reasonable allocation algorithm. If* $m > \beta n$ *then*

$$U_{wc}^{EDF\text{-}RA} \geq n - (n-1)\alpha$$

Proof. Let $\{\tau_1, \ldots, \tau_m\}$ be a set of m tasks which does not fit into the multiprocessor. There are tasks in the set which are allocated to processors, and tasks which are not. Let us change the indexes in the set, so that the tasks which were not allocated have the last indexes in the set. Let τ_k be the first task in the set (after the change of indexes), which was not allocated to any processor. Since the allocation algorithm is reasonable, from (1) we get

$$U_j > 1 - u_k \qquad \text{for } j = 1, \ldots, n \tag{2}$$

where U_j is the total utilization of the tasks previously allocated to processor P_j and u_k is the utilization factor of task τ_k. The total

utilization of the whole set, U, fulfills

$$U = \sum_{i=1}^{m} u_i \geq \sum_{i=1}^{k} u_i = \sum_{j=1}^{n} U_j + u_k \tag{3}$$

From (2) we get

$$\sum_{j=1}^{n} U_j > \sum_{j=1}^{n} (1 - u_k) = n(1 - u_k)$$

Substituting this inequality into (3)

$$U > n(1 - u_k) + u_k = n - (n - 1)u_k$$

From the system definition, all the utilization factors are less than or equal to α, so $u_k \leq \alpha$ and

$$U > n - (n - 1)\alpha$$

Any task set which does not fit into the multiprocessor fulfills the previous expression. Consequently, any task set of total utilization less than or equal to $n - (n - 1)\alpha$ fits into the multiprocessor, and

$$U_{\text{wc}}^{\text{EDF-RA}} \geq n - (n - 1)\alpha$$

□

We have found a lower limit on the utilization bound for any reasonable allocation algorithm. Section 4.1 will present the utilization bound for a reasonable allocation algorithm, which coincides with this lower limit. Therefore, $L_{\text{EDF}}(n, \alpha) = n - (n - 1)\alpha$.

Now we obtain an upper limit on the utilization bound for any allocation algorithm, reasonable or not.

THEOREM 2. *Let AA be an arbitrary allocation algorithm. If* $m > \beta n$ *then*

$$U_{wc}^{EDF\text{-}AA} \leq \frac{\beta n + 1}{\beta + 1}$$

Proof. We will prove that a set of m tasks $\{\tau_1, \ldots, \tau_m\}$ exists, with utilization factors $0 < u_i \leq \alpha$ for all $i = 1, \ldots, m$, and total utilization $U = (\beta n + 1)/(\beta + 1) + \epsilon$, with $\epsilon \to 0^+$, which does not fit into n processors using any allocation algorithm and EDF scheduling on each processor. The proof will be divided into three parts:

1. The task set is presented.

2. The utilization factors of the task set are proved to be valid, that is, $0 < u_i \leq \alpha$.

3. The claim that the task set does not fit into the multiprocessor is proved.

Part 1. The set of m tasks is composed of two subsets: a first subset with $(m - \beta n - 1)$ tasks, and a second subset with $(\beta n + 1)$ tasks.

All the tasks of the first subset have the same utilization factor of value

$$u_i = \frac{\epsilon}{m} \qquad \text{for } i = 1, \ldots, (m - \beta n - 1)$$

All the tasks of the second subset have the same utilization factor of value

$$u_i = \frac{1}{\beta + 1} + \frac{\epsilon}{m} \qquad \text{for } i = (m - \beta n), \ldots, m$$

It is simple to check that the task set is made up of m tasks of total utilization $(\beta n + 1)/(\beta + 1) + \epsilon$.

Part 2. It is also necessary to prove that the utilization factors of both subsets are valid, i.e, $0 < u_i \leq \alpha$ for all $i = 1, \ldots, m$. On one hand, making ϵ low enough,

$$0 < u_i = \frac{\epsilon}{m} < \alpha \qquad \text{for } i = 1, \ldots, (m - \beta n - 1)$$

On the other hand, by definition of β, $(\beta+1)$ tasks of utilization factor α do not fit into one processor, i.e, $(\beta+1)\alpha > 1$. Making ϵ low enough,

$$\alpha > \frac{1}{\beta + 1} + \frac{\epsilon}{m} = u_i \qquad \text{for } i = (m - \beta n), \ldots, m$$

Part 3. There are $(\beta n + 1)$ tasks in the second subset. Hence, at least one processor of the n available should allocate $(\beta+1)$ or more of these tasks. However, no processor can allocate $(\beta+1)$ or more tasks of the second subset, since $(\beta + 1)$ of these tasks have a utilization higher than one in total.

We conclude that the proposed task set of total utilization $(\beta n + 1)/(\beta + 1) + \epsilon$ does not fit into n processors when $\epsilon \to 0^+$, so the utilization bound $U_{\text{wc}}^{\text{EDF-AA}}$ must be less than or equal to $(\beta n+1)/(\beta+1)$.

Remark: the tasks of the first subset are necessary in the proof only to fulfill the restriction of having m tasks. □

We have found an upper limit on the utilization bound for any allocation algorithm, reasonable or not. Section 4.2 will present the

utilization bound for some reasonable algorithms, which coincides with this upper limit. Therefore, $H_{\mathrm{EDF}}(n, \alpha) = (\beta n + 1)/(\beta + 1)$.

4. Utilization bounds for reasonable allocation algorithms

The utilization bound for multiprocessor EDF scheduling depends on the allocation algorithm. In this article, we focus on reasonable allocation algorithms, avoiding impractical allocation algorithms, which would only complicate the mathematical description of the problem.

Below, we present the utilization bounds for EDF multiprocessor scheduling, considering independent and periodic tasks of deadlines equal to periods, and the following allocation algorithms:

- Worst Fit (WF).
- Reasonable allocation Decreasing (RAD). It includes First Fit Decreasing (FFD), Best Fit Decreasing (BFD), Worst Fit Decreasing (WFD), etc.
- First Fit (FF) and Best Fit (BF).
- Other allocation algorithms, including optimal allocation algorithms, First Fit Increasing (FFI), Best Fit Increasing (BFI), and Worst Fit Increasing (WFI).

4.1. Worst fit allocation

The heuristic Worst Fit (WF) allocates tasks sequentially, i.e, one task after another. Each task is allocated to the processor in which it fits the worst. In other words, it is allocated to the processor P_j with the highest free capacity, i.e, highest $(1 - U_j)$, among those with enough capacity to hold the task.

THEOREM 3. *If $m > \beta n$ then*

$$U_{wc}^{EDF\text{-}WF}(n, \alpha) = n - (n - 1)\alpha$$

Proof. Firstly, we will prove the existence of a set of m tasks, $\{\tau_1, \ldots, \tau_m\}$, of utilization factors less than or equal to α, and total utilization

$$n - (n - 1)\alpha + \epsilon$$

with $\epsilon \to 0^+$, which does not fit into the processors using the allocation algorithm WF. The proof will be divided into three parts:

1. The task set is presented.
2. The utilization factors of the task set are proved to be valid, that is, $0 < u_i \leq \alpha$.
3. The claim that the task set does not fit into the multiprocessor is proved.

Part 1. The set of m tasks is built as follows, strictly in the order indicated. A first subset made up of $(m - \beta n - 1)$ tasks of utilization factor

$$u_i = \frac{\epsilon}{2(m - \beta n - 1)} \qquad \text{for } i = 1, \ldots, m - \beta n - 1$$

A second subset made up of βn tasks of utilization factor

$$u_i = \frac{1 - \alpha}{\beta} + \frac{\epsilon}{2\beta n} \qquad \text{for } i = (m - \beta n), \ldots, (m - 1)$$

Finally, the last task has a utilization factor $u_m = \alpha$.

It can be proved that the task set is made up of m tasks, of total utilization $n - (n - 1)\alpha$.

Part 2. It is necessary to prove that the utilization factors of all the tasks are valid, i.e, $0 < u_i \leq \alpha$ for $i = 1, \ldots, m$. Since $\epsilon \to 0^+$, it is enough to prove that $\alpha > (1 - \alpha)/\beta$.

β is calculated from the expression $\beta = \lfloor 1/\alpha \rfloor$, as indicated in Section 3. Therefore, it follows that

$$\beta > \frac{1}{\alpha} - 1 = \frac{1 - \alpha}{\alpha} \quad \text{, and} \quad \frac{1 - \alpha}{\beta} < \alpha$$

Part 3. Next, we will prove that the task set does not fit into the multiprocessor using WF allocation.

The tasks of the first subset fit into the multiprocessor. They are equitatively allocated between all the processors, which means the difference in the number of tasks between any pair of processors is not greater than one.

The βn tasks of the second subset also fit into the multiprocessor. They are equitatively divided into the n processors, i.e, each processor receives β of these tasks. Thus, the total utilization of the tasks allocated to any processor is higher than $(1 - \alpha)$ and the last task, of utilization factor α, does not fit into any processor. Hence, $U_{\text{wc}}^{\text{EDF-WF}} \leq n - (n - 1)\alpha$.

Taking into account the lower limit on the utilization bound for any reasonable allocation algorithm, given by Theorem 1, which includes WF, it follows that

$$U_{\text{wc}}^{\text{EDF-WF}}(n, \alpha) = n - (n - 1)\alpha$$

□

4.2. Reasonable allocation decreasing

The algorithms belonging to the class Reasonable Allocation Decreasing (RAD) order the tasks by decreasing utilization factors before carrying out the allocation (see Section 2). Examples of RAD algorithms are First Fit Decreasing (FFD), Best Fit Decreasing (BFD) and Worst Fit Decreasing (WFD).

All these algorithms share a common utilization bound, as will be proved in Theorem 4. In addition, this common utilization bound coincides with the upper limit given by Theorem 2. Thus, not even the optimal allocation algorithm can provide a higher utilization bound than that of the allocation algorithms in the RAD class.

THEOREM 4. *If $m > \beta n$ then*

$$U_{wc}^{EDF\text{-}RAD}(n,\alpha) = \frac{\beta n + 1}{\beta + 1}$$

Proof. The proof for the case $n = 1$ is trivial. $U_{\text{wc}}^{\text{EDF-RAD}}(n = 1, \alpha) = 1$, so it coincides with the utilization bound for the uniprocessor case, which is independent of the "task sizes". We will assume $n > 1$ in the rest of the proof.

Let $\{\tau_1, \ldots, \tau_m\}$ be a set of m tasks which does not fit into the multiprocessor. Let τ_k be the first task in the set which does not fit into the multiprocessor. Since RAD allocation algorithms are reasonable, from (1) we get

$$U_j > 1 - u_k \qquad \text{for } j = 1, \ldots, n \tag{4}$$

where U_j is the total utilization of the tasks allocated to processor P_j and u_k is the utilization factor of task τ_k. The total utilization of the first k tasks fulfills

$$\sum_{i=1}^{k} u_i = \sum_{j=1}^{n} U_j + u_k \tag{5}$$

From equation (4) we get

$$\sum_{j=1}^{n} U_j > \sum_{j=1}^{n} (1 - u_k) = n(1 - u_k)$$

Substituting this expression into equation (5)

$$\sum_{i=1}^{k} u_i > n(1 - u_k) + u_k = n - (n - 1)u_k \tag{6}$$

Tasks are ordered by decreasing utilization factors before carrying out the allocation, so

$$u_k \leq \frac{\sum_{i=1}^{k} u_i}{k}$$

Substituting this inequality into (6)

$$\sum_{i=1}^{k} u_i > n - (n-1)\frac{\sum_{i=1}^{k} u_i}{k}$$

Finding $\sum_{i=1}^{k} u_i$, it follows that

$$\sum_{i=1}^{k} u_i > \frac{kn}{k+n-1}$$

The total utilization of the first k tasks is less than or equal to the total utilization of the whole task set. Thus,

$$U = \sum_{i=1}^{m} u_i \geq \sum_{i=1}^{k} u_i > \frac{kn}{k+n-1} \tag{7}$$

Parameter k is restricted to $\beta n < k \leq m$. If $\beta n \geq k$, then the first k tasks would fit into the multiprocessor, which would contradict the hypothesis stating that task τ_k does not fit into the multiprocessor. In addition, m is the number of tasks of the set, so $k \leq m$. Next, we will calculate the minimum value of

$$f(k,n) = \frac{kn}{k+n-1} \qquad \text{with } \beta n < k \leq m. \tag{8}$$

Function $f(k,n)$ fulfills

$$\frac{\partial f(k,n)}{\partial k} = \frac{n(n-1)}{(k+n-1)^2} > 0 \qquad \text{for } n > 1$$

Therefore, $f(k,n)$ is increasing in the interval $\beta n < k \leq m$ and its absolute minimum corresponds to $k = (\beta n + 1)$. Substituting k for $(\beta n + 1)$ into (7)

$$U > \frac{\beta n + 1}{\beta + 1}$$

Therefore, a necessary condition to be fulfilled by the total utilization of any task set which does not fit into the n processors is

$$U > \frac{\beta n + 1}{\beta + 1}$$

In other words, any task set of total utilization less than or equal to

$$\frac{\beta n + 1}{\beta + 1}$$

fits into n processors and

$$U_{\text{wc}}^{\text{EDF-RAD}} \geq \frac{\beta n + 1}{\beta + 1}$$

RAD algorithms are reasonable allocation algorithms, so applying Theorem 2

$$U_{\text{wc}}^{\text{EDF-RAD}} \leq \frac{\beta n + 1}{\beta + 1}$$

Therefore, we finally conclude

$$U_{\text{wc}}^{\text{EDF-RAD}}(n, \alpha) = \frac{\beta n + 1}{\beta + 1}$$

□

4.3. First Fit and Best Fit

The heuristic First Fit (FF) allocates tasks sequentially, i.e, one task after another. Each task is allocated to the first processor it fits into. Processors are visited in the order $P_1, \ldots, P_n$.

The heuristic Best Fit (BF) also allocates tasks sequentially. Each task is allocated to the processor into which it fits the best. In other words, it is allocated to the processor P_j with the lowest free capacity, i.e, lowest $(1-U_j)$, among those with enough capacity to hold the task.

The utilization bounds for FF and BF allocation are the same, of value

$$U_{\text{wc}}^{\text{EDF-FF}}(n, \alpha) = U_{\text{wc}}^{\text{EDF-BF}}(n, \alpha) = \frac{\beta n + 1}{\beta + 1}$$

In addition, the proof is almost identical for both heuristics, so we will only prove the multiprocessor utilization bound for FF allocation.

Next, we present the strategy in the proof of the multiprocessor utilization bound for FF allocation.

1. Lemma 1 is proved. This lemma is necessary in order to prove Theorem 5.

2. Theorem 5 proves an expression which relates the utilization for m tasks and n processors, with the utilization bound for $(m-\beta)$ tasks and $(n-1)$ processors. This allows us to prove Theorem 6, going from the case $n = 1$ (uniprocessor case) to a general multiprocessor case with an arbitrary n.

3. From the result of Step 2 and the utilization bound for EDF scheduling on uniprocessors, Theorem 6 obtains the utilization bound for FF allocation.

The proof of Theorem 5 requires Lemma 1, which is proved below. It relates the utilization bounds for the same number of processors, but different number of tasks.

LEMMA 1. *Let $U_{wc}^{EDF\text{-}FF}(m,n,\alpha)$ be the multiprocessor utilization bound for EDF scheduling and FF allocation on n processors of sets of m tasks with utilization factors less than or equal to α. It follows that*

$$U_{wc}^{EDF\text{-}FF}(q,n,\alpha) \geq U_{wc}^{EDF\text{-}FF}(m,n,\alpha) \qquad for\ q < m$$

Proof. Let us consider a task set made up of q tasks which does not fit into the multiprocessor. If we add $(m-q)$ tasks of utilization factors $\epsilon \rightarrow 0^+$ at the end of this task set, nor does the resulting task set fit into n processors. Therefore, the utilization bound for q tasks and n processors can not be lower than the utilization bound for m tasks and n processors. □

Next, we prove an expression which relates the utilization bound of multiprocessors with n and $(n-1)$ processors. This will allow us to obtain a lower limit on the multiprocessor utilization bound, going from the case $n = 1$ (uniprocessor case) to a general multiprocessor case with an arbitrary n.

THEOREM 5. *If $m > \beta n$ then*

$$U_{wc}^{EDF\text{-}FF}(m,n,\alpha) \geq \frac{\beta}{\beta+1} + U_{wc}^{EDF\text{-}FF}(m-\beta, n-1, \alpha)$$

Proof. We will prove that any set of m tasks $\{\tau_1,\ldots,\tau_m\}$, with utilization factors $0 < u_i \leq \alpha$ for all $i = 1,\ldots,m$, and total utilization less than or equal to

$$\frac{\beta}{\beta+1} + U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$$

fits into n processors using EDF scheduling on each processor and FF allocation.
There are two possible cases:

Case 1: The first $(m-\beta)$ tasks have utilization factors less than or equal to $U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$, that is, $\sum_{i=1}^{m-\beta} u_i \leq U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$. In this case, the whole set of m tasks always fits into n processors, because the first $(m-\beta)$ tasks fit into the first $(n-1)$

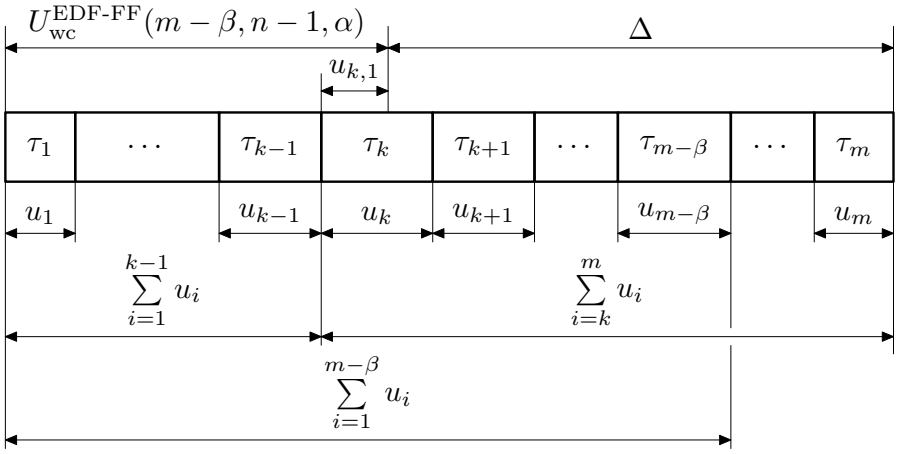

Figure 2. General situation of *case 2* of Theorem 5.

processors (since its utilization is below the bound), and the remaining β tasks fit into the last processor, since the definition of β implies that at least β tasks always fit into one processor.

Case 2: The first $(m-\beta)$ tasks have a total utilization greater than $U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$, that is, $\sum_{i=1}^{m-\beta} u_i > U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$. In this case, we will prove that the whole set of m tasks still fits into n processors if the total utilization is equal to $U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)+\Delta$, provided $\Delta \leq \beta/(\beta+1)$.

A task τ_k must exist, whose u_k added to the previous utilizations u_i, causes the bound $U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$ to be exceeded. This situation is depicted in Figure 2, which is a graphical representation of the utilization factors of each task and the relationships between several quantities and summations used throughout this proof. The value of k is obtained as the integer which fulfills:

$$\sum_{i=1}^{k-1} u_i \leq U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) < \sum_{i=1}^{k} u_i$$

Note that $k \leq m-\beta$, because if $k > m-\beta$ we would be dealing with Case 1.

Let us show that the first $(k-1)$ tasks fit into the first $(n-1)$ processors. The total utilization of the first $(k-1)$ tasks fulfills

$$\sum_{i=1}^{k-1} u_i \leq U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha)$$

Applying Lemma 1 with $(m-\beta) > (k-1)$, we get

$$U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) \leq U_{\text{wc}}^{\text{EDF-FF}}(k-1, n-1, \alpha)$$

and thus

$$\sum_{i=1}^{k-1} u_i \leq U_{\text{wc}}^{\text{EDF-FF}}(k-1, n-1, \alpha)$$

Therefore, the first $(k-1)$ tasks fit into the first $(n-1)$ processors. We have only to prove that the remaining $(m-k+1)$ tasks fit into the last processor.

The worst situation in terms of schedulability appears when all the tasks τ_i in $\{\tau_k, \ldots, \tau_m\}$ fulfill $u_i > u_{k,1}$, where

$$u_{k,1} = U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) - \sum_{i=1}^{k-1} u_i$$

as shown in Figure 2. Note that if there is a task τ_i in $\{\tau_k, \ldots, \tau_m\}$ with $u_i \leq u_{k,1}$, we can always allocate this task to the first $(n-1)$ processors (since the addition of this new task does not cause the total utilization to exceed the bound), and the situation is analogous to the current one, with k one unit greater. This reasoning can be repeated until no task τ_i with $u_i \leq u_{k,1}$ exists among the last $(m-k+1)$ tasks.

In order to prove that the last $(m-k+1)$ tasks fit into the last processor we have to prove that the total utilization of these tasks is not greater than one, that is,

$$\sum_{i=k}^{m} u_i \leq 1$$

Figure 2 shows that

$$\sum_{i=k}^{m} u_i = u_{k,1} + \Delta \tag{9}$$

As already stated, all the utilization factors u_i in this summation are greater than $u_{k,1}$, so

$$\begin{aligned}(m-k+1)u_{k,1} &< u_{k,1} + \Delta \\ &\leq u_{k,1} + \frac{\beta}{\beta+1} \qquad \text{by the definition of } \Delta\end{aligned}$$

and we can find $u_{k,1}$.

$$u_{k,1} < \frac{\beta}{(\beta+1)(m-k)} \tag{10}$$

Substituting the value of $u_{k,1}$ from (10) into (9) we obtain

$$\begin{aligned}\sum_{i=k}^{m} u_i &< \frac{\beta}{(\beta+1)(m-k)} + \Delta \\ &< \frac{\beta}{(\beta+1)(m-k)} + \frac{\beta}{(\beta+1)} \qquad \text{by def. of } \Delta \\ &= \frac{(m-k+1)\beta}{(\beta+1)(m-k)} \\ &= \frac{1+1/(m-k)}{1+1/\beta}\end{aligned}$$

We know that $k \leq m - \beta$ in case 2, so $m - k \geq \beta$. Therefore,

$$\sum_{i=k}^{m} u_i \leq 1$$

This equation shows that the last $(m - k + 1)$ tasks meet the EDF uniprocessor schedulability condition, so they fit into the last processor.

We have proved that any task set with m tasks and a total utilization

$$U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) + \Delta \leq U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) + \frac{\beta}{\beta+1}$$

fits into n processors. Therefore, the utilization bound $U_{\text{wc}}^{\text{EDF-FF}}(m, n, \alpha)$, must be greater than or equal to $U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) + \beta/(\beta+1)$, and the theorem is proved. □

Theorem 5 is also valid for BF allocation. Nevertheless, proving the statement "the worst situation in terms of schedulability appears when all the tasks τ_i in $\{\tau_k, \ldots, \tau_m\}$ fulfill $u_i > u_{k,1}$", requires some elaboration for BF allocation and is not shown for the sake of brevity.

The utilization bound for FF allocation is obtained in Theorem 6.

THEOREM 6. *If $m > \beta n$ then*

$$U_{wc}^{EDF\text{-}FF}(n, \alpha) = \frac{\beta n + 1}{\beta + 1} \tag{11}$$

Proof. First, we obtain a lower limit on the utilization bound for a set of m tasks on a multiprocessor with n processors, using FF allocation.

Theorem 5 relates the utilization bound of sets of m tasks on multiprocessors of n processors, with utilization bound of sets of $(m-\beta)$ tasks on multiprocessors with $(n-1)$ processors.

$$U_{\text{wc}}^{\text{EDF-FF}}(m,n,\alpha) \geq \frac{\beta}{\beta+1} + U_{\text{wc}}^{\text{EDF-FF}}(m-\beta, n-1, \alpha) \tag{12}$$

Theorem 5 also relates the utilization bound of sets of $(m-\beta)$ tasks on multiprocessors of $(n-1)$ processors, with the utilization bound of sets of $(m-2\beta)$ tasks on multiprocessors of $(n-2)$ processors.

$$U_{\text{wc}}^{\text{EDF-FF}}(m-\beta,n-1,\alpha) \geq \frac{\beta}{\beta+1} + U_{\text{wc}}^{\text{EDF-FF}}(m-2\beta, n-2, \alpha) \tag{13}$$

Substituting (13) into (12) we get

$$U_{\text{wc}}^{\text{EDF-FF}}(m,n,\alpha) \geq \frac{2\beta}{\beta+1} + U_{\text{wc}}^{\text{EDF-FF}}(m-2\beta, n-2, \alpha)$$

This procedure can be repeated, until finally relating the utilization bound of sets of m tasks on multiprocessors of n processors, with the utilization bound of sets of $(m-(n-1)\beta)$ tasks on a uniprocessor.

$$U_{\text{wc}}^{\text{EDF-FF}}(m,n,\alpha) \geq \frac{(n-1)\beta}{\beta+1} + U_{\text{wc}}^{\text{EDF-FF}}(m-(n-1)\beta, 1, \alpha) \tag{14}$$

The utilization bound for $(m-(n-1)\beta)$ tasks and one processor is one, which does not depend on the values of m or α.

$$U_{\text{wc}}^{\text{EDF-FF}}(m-(n-1)\beta,1,\alpha) = 1 \tag{15}$$

Substituting (15) into (14), we obtain a lower limit on the utilization bound of m tasks on n processors.

$$U_{\text{wc}}^{\text{EDF-FF}}(m,n,\alpha) \geq \frac{\beta n+1}{\beta+1}$$

Theorem 2 proved that

$$U_{\text{wc}}^{\text{EDF-RA}}(m,n,\alpha) \leq \frac{\beta n+1}{\beta+1}$$

and FF is a reasonable allocation algorithm. Hence,

$$U_{\text{wc}}^{\text{EDF-FF}}(m,n,\alpha) = \frac{\beta n+1}{\beta+1}$$

From this expression we observe that $U_{\text{wc}}^{\text{EDF-FF}}$ does not depend on the number of tasks, m, so our final conclusion is that

$$U_{\text{wc}}^{\text{EDF-FF}} = U_{\text{wc}}^{\text{EDF-FF}}(n, \alpha) = \frac{\beta n + 1}{\beta + 1}$$

□

4.4. Other allocation algorithms

In this section we consider optimal allocation algorithms and the heuristics First Fit Increasing (FFI), Best Fit Increasing (BFI) and Worst Fit Increasing (WFI).

An optimal allocation algorithm (OPT) is one able to allocate any task set whenever a feasible allocation exists.

The heuristics FFI, BFI and WFI are variants of the heuristics FF, BF and WF respectively. They order the tasks by increasing utilization factors before carrying out the allocation. Therefore, the task with the lowest utilization factor (the lowest "size") is allocated first.

The utilization bound for an optimal allocation algorithm coincides with H_{EDF}, so it also coincides with the utilization bound for FF, BF and RAD allocation. Therefore,

$$U_{\text{wc}}^{\text{EDF-OPT}}(n, \alpha) = \frac{\beta n + 1}{\beta + 1}$$

The utilization bound for WFI allocation coincides with that of WF allocation. The task set defined in Theorem 3, in Section 4.1, was made up of tasks ordered by increasing utilization factors and the task set was proved not to fit into the multiprocessor using WF allocation. Therefore,

$$U_{\text{wc}}^{\text{EDF-WFI}}(n, \alpha) = U_{\text{wc}}^{\text{EDF-WF}}(n, \alpha) = n - (n-1)\alpha$$

Let us calculate now the utilization bound for FFI allocation. For any task set that can not be allocated using FFI allocation, there is another task set with the same total utilization that can not be allocated using FF allocation. This last task set can be obtained by ordering the tasks of the first task set by increasing utilization factors. Therefore, the utilization bound for FF allocation can not be higher than the utilization bound for FFI allocation, i.e, $U_{\text{wc}}^{\text{EDF-FFI}} \geq U_{\text{wc}}^{\text{EDF-FF}}$. Since the utilization bound for FF allocation coincides with the maximum utilization bound, H_{EDF}, it follows that

$$U_{\text{wc}}^{\text{EDF-FFI}}(n, \alpha) = U_{\text{wc}}^{\text{EDF-FF}}(n, \alpha) = \frac{\beta n + 1}{\beta + 1}$$

Applying a similar reasoning for BFI allocation,

$$U_{\text{wc}}^{\text{EDF-BFI}}(n,\alpha) = U_{\text{wc}}^{\text{EDF-BF}}(n,\alpha) = \frac{\beta n + 1}{\beta + 1}$$

5. Analysis of the theoretical results

All the utilization bounds for multiprocessor EDF scheduling presented in the previous section fall into two categories:

- The utilization bounds for WF and WFI coincide with the minimum utilization bound, given by

$$L_{\text{EDF}}(n,\alpha) = n - (n-1)\alpha = n(1-\alpha) + \alpha$$

- The utilization bounds for FF, BF, FFI, BFI and the class RAD coincide with the maximum utilization bound, given by

$$H_{\text{EDF}}(n,\alpha) = \frac{\beta n + 1}{\beta + 1}$$

Therefore, we will analyze only the expressions of L_{EDF} and H_{EDF}.

Figure 3 depicts the function $L_{\text{EDF}}(n,\alpha)$ as a function of the number of processors, for different values of α. Although n is an integer, it is represented as a continuous function with the aim of improving its visualization. The representation is normalized by dividing L_{EDF} by the number of processors, in order to show the average degree of utilization of the processors.

$L_{\text{EDF}}(1,\alpha) = 1$, which corresponds to the bound for uniprocessor EDF scheduling. Every time we add a new processor to the system, the utilization bound is increased by $(1-\alpha)$.

For high values of α, the utilization bound L_{EDF} is too small to be practical. It is close to 1.0, so L_{EDF}/n nears 0, for any number of processors. However, as α nears 0, the utilization bound L_{EDF} becomes close to n, so L_{EDF}/n nears 1.0. In this case, the multiprocessor behaves approximately like a uniprocessor n times faster.

Figure 4 depicts the function $H_{\text{EDF}}(n,\alpha)$ as a function of the number of processors for different values of α. Although n is an integer, it is again represented as a continuous function to improve its visualization. The representation is normalized by dividing H_{EDF} by the number of processors. Each curve in Figure 4 corresponds to a different value of β, and therefore to different values of α.

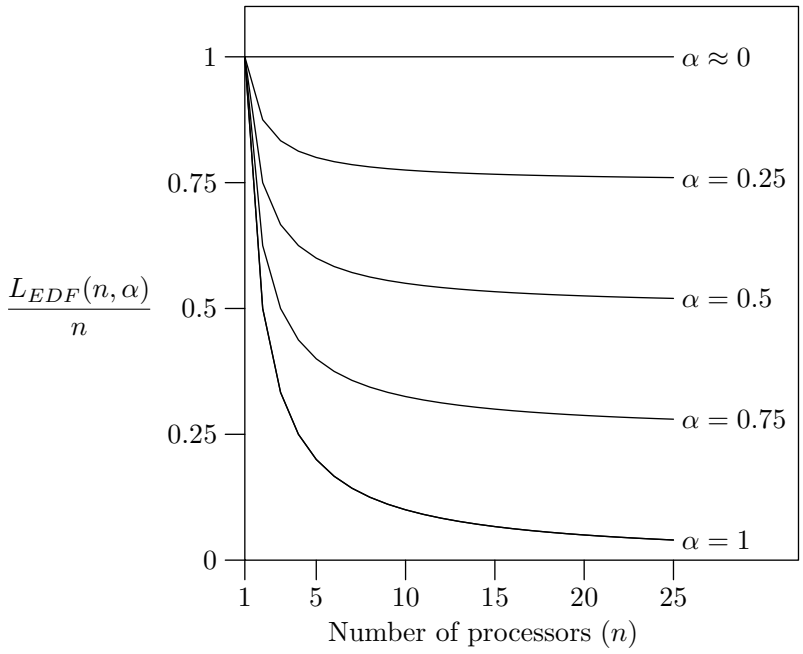

Figure 3. Plot of $L_{\mathrm{EDF}}(n, \alpha)$.

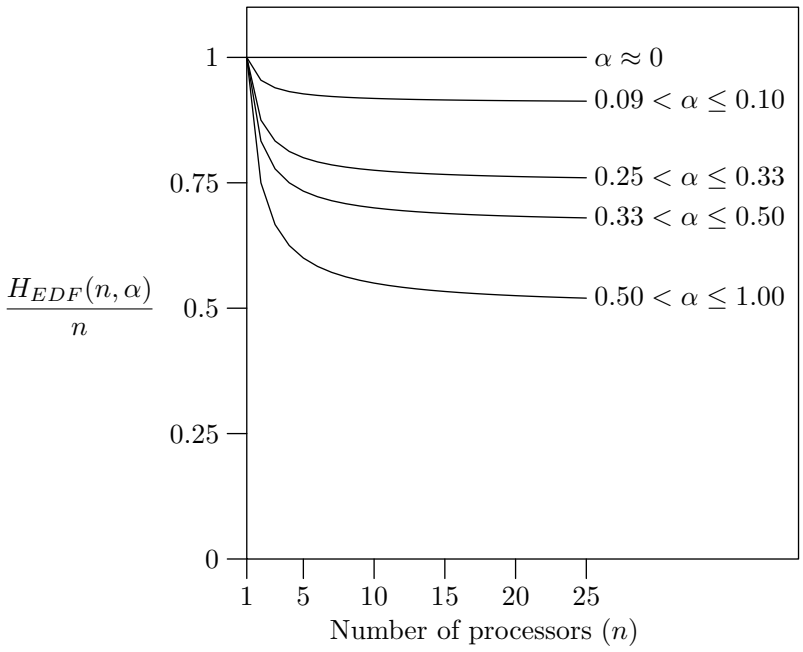

Figure 4. Plot of $H_{\mathrm{EDF}}(n, \alpha)$.

$H_{\mathrm{EDF}}(1, \alpha) = 1$, which corresponds to the bound for uniprocessor EDF scheduling. Every time we add a new processor to the system, the utilization bound is increased by $\beta/(\beta + 1)$.

For $\alpha > 0.5$, the utilization bound H_{EDF} is equal to $0.5(n + 1)$. As α nears 0, the utilization bound H_{EDF} becomes close to n and H_{EDF}/n close to 1.0. In this case, the multiprocessor behaves approximately like a uniprocessor n times faster.

For example, the utilization bound associated with multiprocessor EDF scheduling and FF allocation in a multiprocessor made up of two processors is 1.5, i.e, 0.75 per processor. If the tasks have utilization factors less than or equal to 0.25, then $\beta = 4$. In this case, the utilization bound for FF allocation takes the value 1.8, i.e, 0.9 per processor, close to the ideal 1.0 per processor.

Finally, it should be pointed out that the performance of any allocation algorithm is not defined only by its utilization bound. Utilization bounds consider only the worst-case. For example, any optimal allocation algorithm has the same EDF multiprocessor utilization bound as FFD allocation. However, optimal allocation algorithms are able to allocate task sets which can not be allocated using FFD allocation.

6. Task model extensions

All the results concerning multiprocessor utilization bounds provided in Section 4 were obtained using a basic task model, made up of periodic and independent tasks of deadlines equal to periods.

The objective of this section is to provide utilization bounds for more complex task sets. In some cases, the utilization bounds will be tight and in other pessimistic.

Each of the following subsections introduces a different extension with regard to the basic task model. In any case, the results of all the subsections may be merged to cover simultaneously all the extensions.

6.1. Arbitrary deadlines

The multiprocessor utilization bounds for EDF scheduling provided in Section 4 are valid even when some deadlines are higher than the periods. The multiprocessor EDF utilization bounds depend on the uniprocessor EDF utilization bounds, which are 1.0 for deadlines equal to periods and for deadlines higher than the periods.

Let us consider the case of deadlines less than the periods. Assuming that $D_i = \Delta T_i$ for all the tasks, with $\Delta \leq 1$, the new utilization bound

for EDF uniprocessor scheduling becomes

$$U^{\mathrm{EDF}} = \Delta$$

Any task set Γ with utilization factors $\{u_1, \ldots, u_m\}$ and deadlines $\{\Delta T_1, \ldots, \Delta T_m\}$ fits into the multiprocessor, if and only if, the task set Γ' with utilization factors $\{u_1/\Delta, \ldots, u_m/\Delta\}$ and deadlines $\{T_1, \ldots, T_m\}$ fits into the multiprocessor. The proof is simple: the capacity of the processor for the task set Γ' is 1, i.e, $(1/\Delta)$ times higher than the capacity for task set Γ, but the utilization factors of Γ' are also $(1/\Delta)$ times greater than those of Γ.

The multiprocessor utilization bound coincides with the total utilization of the worst-case task set (minus an ϵ), the worst-case task set being the one with the lowest utilization among those which do not fit into the multiprocessor. Therefore, the worst-case task set for $\Delta < 1$ can be obtained by multiplying the utilization factors of the worst-case task set for $\Delta = 1$ by the term Δ. Thus, the utilization bounds for $\Delta < 1$ are obtained by multiplying the utilization bounds of the previous sections by Δ.

To sum up, the utilization bounds for the case $\Delta < 1$ can be obtained following these steps:

1. Transform the original task set, Γ, with $\Delta < 1$, into a new task set, Γ', with $\Delta = 1$, by multiplying the utilization factors and deadlines by $(1/\Delta)$. Therefore, it follows that $\alpha' = \alpha/\Delta$, where α' is the maximum reachable utilization factor of Γ', and $\beta' = \lfloor 1/\alpha' \rfloor = \lfloor \Delta/\alpha \rfloor$. Note that if β' is zero, Γ' may have tasks with utilization factors greater than 1, and the task set is not schedulable.

2. Calculate the multiprocessor utilization bounds using β' instead of β.

3. Multiply the utilization bound obtained for Γ' by Δ to obtain the utilization bound for the original task set, Γ.

For example, the multiprocessor utilization bound for FF allocation and $\Delta = 0.5$ becomes $0.5(\beta' n + 1)/(\beta' + 1)$, with $\beta' = \lfloor 0.5/\alpha \rfloor$

The condition $D_i = \Delta T_i$ is not common in practice. However, the multiprocessor utilization bounds assuming this condition give us insight into the schedulability variations against deadlines.

The same technique could be applied to any uniprocessor scheduling algorithm providing a fixed uniprocessor utilization bound. For example, a pessimistic (but close to the tight) utilization bound for uniprocessor RM scheduling is $\ln 2$. Therefore, a pessimistic (but close to the tight) multiprocessor utilization bound for RM scheduling and FF allocation is $\ln 2(\beta' n + 1)/(\beta' + 1)$, with $\beta' = \lfloor \ln 2/\alpha \rfloor$.

6.2. Release jitter

The basic task model assume that the interarrival times of periodic tasks are strictly constant, equal to the periods. However, it is common to find some delay in the release times of the tasks called jitter (Tindell and Clark, 1994). Jitter is a source of unpredictability, which should be bounded in some way. Frequently, a maximum jitter J_i is associated to each task τ_i. Therefore, if a job of a periodic task τ_i is theoretically released at time $\lambda_{i,j}$, in practice the release time may be any instant in $[\lambda_{i,j}, \lambda_{i,j} + J_i]$.

The jitter effect on the EDF uniprocessor bound can be taken into account in a simple way if we admit some pessimism. If we artificially increase the computation time of all the tasks by their jitters, the schedulability of the resulting task set (without jitter) implies the schedulability of the original task set (with jitter). The proof is not included for the sake of brevity.

Therefore, the multiprocessor utilization bounds for EDF scheduling of periodic tasks without jitter are totally valid for task sets with jitter, increasing the computation times of the tasks by their jitters.

Obviously this produces some pessimism, acceptable for low values of jitter.

6.3. Context switches

The cost of context switches should be accounted for in any realistic schedulability analysis. In the worst case, each time a job is released or completed, a context switch may occur. Therefore, the cost of context switches can be easily included in the analysis by adding the computational cost of performing two context switches to the computation time of all the tasks. Since the computational cost of context switches is usually much lower than the computation times of the tasks, the pessimism introduced into the analysis is usually tolerable.

Therefore, multiprocessor utilization bounds are valid by simply increasing the computation times of the tasks by twice the cost of a context switch.

6.4. Aperiodic tasks

Aperiodic tasks are characterized by soft deadlines and unpredictable interarrival times. The objectives of systems running periodic and aperiodic tasks together is to reduce the average response time of aperiodic tasks without jeopardizing the schedulability of periodic tasks.

In order to meet these objectives, aperiodic tasks are scheduled using aperiodic servers (Buttazzo, 1997; Bernat and Burns, 1999). Depending on the aperiodic server, the interference with periodic tasks is different.

Those aperiodic servers that interfere on periodic tasks strictly like an equivalent periodic task, allow us to apply the utilization bounds provided in Section 4. We need only substitute the aperiodic servers by their equivalent periodic tasks and ignore the aperiodic tasks. Examples of these servers can be found in Buttazzo (1997).

Sporadic tasks are a special case of aperiodic tasks with interarrival times bounded by a minimum value, and they are usually characterized by hard deadlines. In the worst case, any sporadic task behaves like a periodic task of period equal to its minimum interarrival time. Thus, assuming some pessimism, the multiprocessor utilization bounds for task sets including sporadic tasks are identical to those provided for periodic task sets.

6.5. Shared resources

The basic task model assumes independent tasks. However, tasks become dependent when they communicate through shared resources, like shared memory areas.

Access to shared resources is controlled by protocols that avoid data inconsistencies and limit the blocking time. One of these protocols for uniprocessors is the Stack Resource Protocol (SRP). Assuming some pessimism, the uniprocessor schedulability of a task set using EDF and this protocol can be tested by increasing the total utilization by the term $\max_{\tau_i}(B_i/T_i)$, where B_i is the maximum blocking time of task τ_i (Baker, 1991).

$$\sum u_i + \max_{\tau_i}(B_i/T_i) \leq 1 \tag{16}$$

We will assume that all the processors use EDF scheduling and the SRP protocol to access any shared resource. All the tasks linked each other by shared resources make what we call a macrotask. In order to clarify the concepts of dependency and macrotask, we will refer to Figure 5. Tasks τ_1, τ_2 and τ_3 share a resource. In addition, task τ_4 shares another resource with τ_3. These four tasks are dependent (directly or indirectly) and make macrotask γ_1. Task τ_5 makes macrotask γ_2. Tasks τ_6 and τ_7 share a resource and make macrotask γ_3. Note that all the macrotasks are independent.

The utilization bounds for tasks, obtained in Section 4, can be applied to macrotasks. A set of m dependent tasks $\{\tau_i\}$ is transformed into a set of $\hat{m}$ independent macrotasks $\{\gamma_k\}$ of utilization factors $\hat{u}_k$,

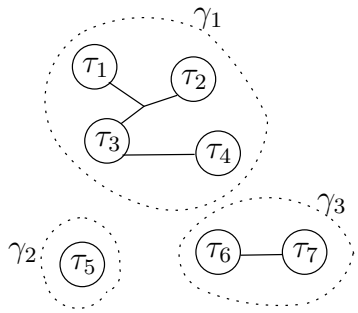

Figure 5. Concepts of dependency and macrotask.

given by equation (17).

$$\hat{u}_k = \sum_{\tau_i \in \gamma_k} u_i + \max_{\tau_i \in \gamma_k}(B'_i/T_i) \tag{17}$$

B'_i is the maximum blocking time of τ_i assuming that all the tasks in the system were allocated to the same processor. If B_i is the maximum blocking time of τ_i considering only the tasks allocated to the same processor as τ_i, it follows $B_i \leq B'_i$. This is because adding more tasks increases the number of possible blockings.

We have transformed a set of dependent tasks into a set of independent macrotasks. The schedulability of the macrotask set on the multiprocessor implies the schedulability of the task set on the multiprocessor. Next we present a proof of this claim.

THEOREM 7. *If a macrotask set $\{\gamma_k\}$ can be allocated to a multiprocessor using a given allocation algorithm, the task set $\{\tau_i\}$, from which the macrotask set was derived, can also be allocated using the same allocation algorithm.*

Proof. If a macrotask set is schedulable, the $\hat{m}$ macrotasks of utilization factors $\{\hat{u}_1, \ldots, \hat{u}_{\hat{m}}\}$ can be allocated to the multiprocessor. Let us consider any processor P_j in the multiprocessor, which receives $\hat{m}_j$ macrotasks. It follows that $\sum_{k=1}^{\hat{m}_j} \hat{u}_k \leq 1$. Therefore, from equation (17) we get

$$\sum_{k=1}^{\hat{m}_j} \sum_{\tau_i \in \gamma_k} u_i + \sum_{k=1}^{\hat{m}_j} \max_{\tau_i \in \gamma_k}(B'_i/T_i) \leq 1$$

The previous equation is equivalent to

$$\sum_{\tau_i \in P_j} u_i + \sum_{k=1}^{\hat{m}_j} \max_{\tau_i \in \gamma_k}(B'_i/T_i) \leq 1 \tag{18}$$

Table I. Example of dependent task set.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
u_i	0.1	0.2	0.2	0.1	0.4	0.1	0.2
B'_i/T_i	0.05	0.03	0.02	0	0	0.1	0.03

Since $B'_i \geq B_i$, it follows that

$$\max_{\tau_i \in \gamma_k}(B'_i/T_i) \geq \max_{\tau_i \in \gamma_k}(B_i/T_i) \tag{19}$$

In addition,

$$\sum_{k=1}^{\hat{m}_j} \max_{\tau_i \in \gamma_k}(B'_i/T_i) \geq \max_{\tau_i \in P_j}(B'_i/T_i) \tag{20}$$

Thus, from equations (18), (19) and (20), we finally conclude

$$\sum_{\tau_i \in P_j} u_i + \max_{\tau_i \in P_j}(B_i/T_i) \leq 1$$

i.e, equation (16) is fulfilled and the set of tasks making up the macrotasks allocated to processor P_j fits into P_j. □

For example, let us consider the set of periodic and dependent tasks of Figure 5, which are defined by the parameters of Table I.

We want to know if the task set is schedulable on a multiprocessor made up of two processors using EDF scheduling and FF allocation.

There are three macrotasks, of utilization factors $\hat{u}_1$, $\hat{u}_2$ and $\hat{u}_3$.

$$\hat{u}_1 = 0.1 + 0.2 + 0.2 + 0.1 + \max\{0.05, 0.03, 0.02\} = 0.65$$
$$\hat{u}_2 = 0.4 + \max\{0\} = 0.4$$
$$\hat{u}_3 = 0.1 + 0.2 + \max\{0.1, 0.03\} = 0.4$$

Parameter α is set at 0.65, the maximum of $\hat{u}_k$, giving $\hat{\beta} = \lfloor 1/0.65 \rfloor = 1$. The multiprocessor utilization bound is

$$U_{\text{wc}}^{\text{EDF-FF}}(n, \hat{\beta}) = \frac{\hat{\beta}n + 1}{\hat{\beta} + 1} = 1.5 \geq (0.65 + 0.4 + 0.4)$$

Therefore, the task set is schedulable.

The reader should note that the concept of macrotasks allow us to deal with task that share resources, although it introduces some pessimism.

6.6. Non-preemptive sections

The system model presented in Section 2 assumes preemption. This means, if one job of high priority is released while a low priority task is executing, a context switch happens, moving the low priority job to the pending queue and executing the high priority job.

Nevertheless, there are situations in which it is necessary to execute a set of instructions atomically, i.e, without any kind of interruption. The set of instructions executed atomically is called a non-preemptive section. The atomic execution can be accomplished by disabling interrupts at the beginning of the non-preemptive section and enabling interrupts at the end. A non-preemptive section can be part of the computation of a job and invalidate the previous analysis, since the response times of the tasks may be higher than expected.

Let us focus on non-preemptive sections included in the computation of a low priority job, i.e, with a late absolute deadline under EDF. If the low priority job is executing within its non-preemptive section and a high priority job is released, the high priority job will suffer blocking. In the worst case, the blocking will be equal to the length of the non-preemptive section. The situation is analogous to that of blocking while accessing protected shared resources. However, there is a basic difference: any job can be blocked at one non-preemptive section at most, and no protocol is necessary to enforce this property.

The multiprocessor utilization bounds provided in Section 6.5 can be applied to the case of non-preemptive sections by increasing the blocking time B'_i of each task τ_i by the longest non-preemptive section of all the tasks except τ_i. Note that non-preemptive sections in τ_i are not considered, because a task can not block itself.

This approach is pessimistic due to many factors, e.g., in the calculation of the maximum blocking coming from non-preemptive sections we are assuming that all the tasks are allocated to the same processor. However, if non-preemptive sections are short, as in general they should be, the pessimism introduced in the analysis is tolerable.

6.7. Mode changes

There are situations in which the environment of a real-time system progresses through well defined states, each state requiring a different task set. This scenario is called mode changes in the literature (Pedro and Burns, 1998).

Any mode change divide the timeline into three parts:

- A steady part before the mode change occurs.
- A transient part during the mode change.

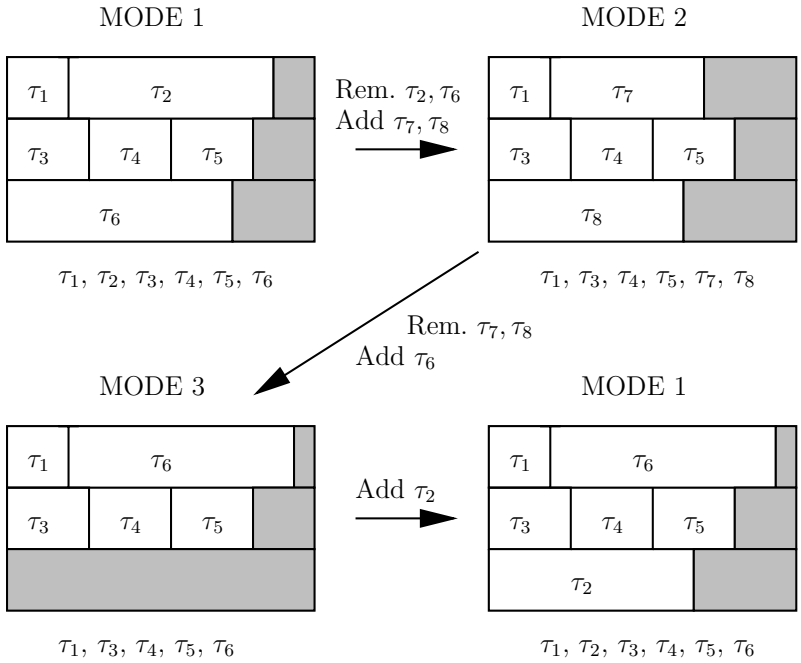

Figure 6. Example of mode changes on multiprocessors.

– A steady part after the mode change.

Independently of the tasks involved in the mode change: old mode completed, old mode aborted, wholly new, unchanged new tasks or changed new tasks (Pedro and Burns, 1998); the steady parts before and after any mode change can be described by a process of removing and adding tasks. Every time a task is removed it leaves space in one processor. Every time a new task is added to the system, it should be allocated to one processor. The rest of the tasks can not migrate to other processors, since it would require stopping and restarting them again. Therefore, allocation algorithms such as FFD, which imply task ordering, can not be applied to dynamic situations such as changes. Quite the opposite, allocation algorithms such as FF, which do not require task ordering, can be applied to mode changes.

Figure 6 shows an example of three mode changes on a multiprocessor using EDF scheduling and FF allocation. Each task is represented by a box of width equal to its utilization factor. Free space in each processor is represented by a shaded area. The reader should observe that the system starts at Mode 1 with one allocation and ends up at the same mode with the same tasks, but allocated differently.

On one hand, independently of the reasonable allocation algorithm we use, the multiprocessor utilization bound can not be higher than L_{EDF}. The process of task removing during a mode change can draw the system to the worst situation, presented in the proof of Theorem 3 for WF allocation. For example, let us consider a multiprocessor made up of two identical processors with EDF scheduling. Let us consider an initial mode defined by four tasks $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ of utilization factors $\{\epsilon, (1-\epsilon), \epsilon, (1-\epsilon)\}$. These tasks are allocated using FF allocation. Each processor allocates one task of utilization factor ϵ and another task of utilization factor $(1-\epsilon)$. Let us assume a mode change consisting of removing the tasks of utilization factor $(1-\epsilon)$ and adding a new task of utilization factor $(1-0.5\epsilon)$. This last task can not be allocated to any processor in the new mode. The situation coincides with the worst situation for WF allocation. The total utilization of the task set in the new mode is $(1+1.5\epsilon)$ which is below the utilization bound 1.5 for FF allocation on two processors, but not below the utilization bound 1.0 for WF allocation on two processors.

On the other hand, independently of the reasonable allocation algorithm we use, the multiprocessor utilization bound can not be lower than L_{EDF}, since the task allocation at any mode could be generated using a reasonable allocation algorithm.

Therefore, we conclude that the multiprocessor utilization bound for any reasonable allocation algorithm under mode changes coincides with $L_{\mathrm{EDF}}(n, \alpha) = n - (n-1)\alpha$.

This utilization bound provides a global schedulability condition independent of the task allocation. If we tried to apply local schedulability conditions in each processor, such as those based on response times, we would have to analyze the schedulability for each processor for all the possible allocations in each mode. For example, Figure 6 shows two different task allocations for Mode 1. After a second repetition of the cycle Mode 1→Mode 2→Mode 3→Mode 1, the task allocation for the final Mode 1 may be different. Since the number of possible allocations for the same mode may be huge, local schedulability conditions are not applicable.

Note that the utilization bound for mode changes is only valid in the steady states, i.e, at times far enough from the instant of the mode change. Overloads are possible close to the mode change instant (Pedro and Burns, 1998), so the steady state utilization bound L_{EDF} can not be applied. In addition, if there is no restriction on the utilization factor of the tasks, then $\alpha = 1$ and the utilization bound is 1.0, too low to be useful in practice.

7. Conclusions and future work

The uniprocessor utilization bound for EDF scheduling on uniprocessors has been extended to multiprocessors under a partitioning strategy and arbitrary (reasonable) allocation algorithms. The bound depends on the number of processors and the "task sizes", limited by α, but does not depend on the number of tasks. For the case of tasks with low utilization factors, the utilization bound is greatly raised, asymptotically reaching the ideal value n, when all the utilization factors are close to zero ($\alpha \approx 0$).

In general, the calculation of utilization bounds is a problem of importance real-time systems theory. Utilization bounds allow us not only to perform fast schedulability tests, but also to perform a schedulability analysis. That is, utilization bounds allow us to establish the influence of different parameters such as the number of tasks, task size, etc, on the schedulability of the system by considering the worst-case. Utilization bounds indicate how far the system is from the ideal situation, in which the total utilization equals the number of processors in the system. Furthermore, multiprocessor utilization bounds allow us to perform global schedulability tests for the whole multiprocessor. This is useful in complex scenarios, like mode changes.

We have proved that the multiprocessor utilization bound for any (reasonable) allocation algorithm is in the interval

$$\left[n - (n-1)\alpha, \quad \frac{\lfloor 1/\alpha \rfloor \, n + 1}{\lfloor 1/\alpha \rfloor + 1} \right]$$

When all the utilization factors near zero, $\alpha \approx 0$, and the EDF multiprocessor utilization bound for any allocation algorithm is n.

The multiprocessor utilization bound for First Fit, First Fit Increasing, Best Fit, Best Fit Increasing and optimal allocation algorithms coincide with the maximum. Furthermore, all the (reasonable) allocation algorithms that order the tasks by decreasing utilization factors before carrying out the allocation, have a multiprocessor utilization bound equal to the maximum. This is the case of the allocation algorithms First Fit Decreasing, Best Fit Decreasing and Worst Fit Decreasing.

The utilization bound for Worst Fit and Worst Fit Increasing allocation coincides with the minimum. When there is no restriction on the utilization factors of the tasks, then $\alpha = 1$ and this minimum becomes 1.0 for any number of processors.

All multiprocessor utilization bounds for the basic task model were extended to deal with complex tasks, including aperiodic tasks, arbitrary deadlines, release jitter, mode changes, context switches, non-preemptive sections and blocking in shared resources.

Future work will address the extension of the utilization bounds to distributed real-time systems through the use of jitter. One of the problems is the pessimism of the jitter extension provided in Section 6.2, which is useful for low values of jitter, but too pessimistic for high values of jitter. In addition, an integral analysis of the communications network and processors is necessary (Tindell et al., 1994), which may require obtaining two interrelated utilization bounds, one for the communications network and another for the processors.

References

Baker, T.: 1991, 'Stack-Based scheduling of Real-Time Processes'. *Real-Time Systems* **3**(1), 301–324.

Bernat, G. and A. Burns: 1999, 'New Results on Fixed Priority Aperiodic Servers'. In: *Proceedings of the Real-Time Systems Symposium*.

Burchard, A., J. Liebeherr, Y. Oh, and S. Son: 1995, 'New Strategies for Assigning Real-Time Tasks to Multiprocessor Systems'. *IEEE Transactions on Computers* **44**(12).

Buttazzo, G.: 1997, *Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications*, Chapt. 7. Boston/Dordrecht/London: Kluwer Academic Publishers.

Dall, S. and C. Liu: 1978, 'On a Real-Time Scheduling Problem'. *Operations Research* **6**(1), 127–140.

Davari, S. and S. Dhall: 1986a, 'On a Periodic Real-Time Task Allocation problem'. In: *Annual international Conference on Systems Sciences*. pp. 133–141.

Davari, S. and S. Dhall: 1986b, 'An on Line Algorithm for Real Time Tasks Allocation'. In: *Proceedings of the IEEE Real-Time Systems Symposium*. pp. 194–200.

Dertouzos, M. L.: 1974, 'Control Robotics: The procedural Control of Physical Processes'. *Proceedings of IFIP Congress* pp. 807–813.

Dertouzos, M. L. and A. K. Mok: 1989, 'Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks'. *Transactions on Software Engineering* **15**(12), 1497–1506.

Garey, M. and D. Johnson: 1979, *Computers and Intractability*. New York: W.H. Freman.

Lauzac, S., R. Melhem, and D. Mossé: 1998, 'An Efficient RMS Admission Control and its Application to Multiprocessor Scheduling'. In: *Proceedings of the International Parallel Processing Symposium*. pp. 511–518.

Liu, C. and J. Layland: 1973, 'Scheduling Algorithms for Multiprogramming in a Hard-Real-time Environment'. *Journal of the ACM* **20**(1), 46–61.

López, J., J. Díaz, M. García, and D. García: 2003, 'Utilization Bounds for Multiprocessor Rate-Monotonic Scheduling'. *Real-Time Systems* **24**(1), 5–28.

Oh, D. and T. Baker: 1998, 'Utilization Bounds for N-Processor Rate Monotone Scheduling with Static Processor Assignment'. *Real-Time Systems* **15**(2), 183–193.

Oh, Y. and S. Son: 1995, 'Allocating Fixed-Priority Periodic Tasks on Multiprocessor Systems'. *Real-Time Systems* **9**(3), 207–239.

Pedro, P. and A. Burns: 1998, 'Schedulability Changes for Mode Changes in Flexible Real-Time Systems'. In: *Proceedings of the Euromicro Workshop on Real Time Systems*. pp. 172–179.

Sáez, S., J. Vila, and A. Crespo: 1998, 'Using Exact Feasibility Tests for Allocating Real-Time Tasks in Multiprocessor Systems'. *Proceedings of the 10th Euromicro Workshop on Real-Time Systems* pp. 53–60.

Tindell, K., A. Burns, and A. Wellings: 1994, 'An extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks'. *Real-Time Systems* **6**(2), 133–151.

Tindell, K. and J. Clark: 1994, 'Holistic Schedulability Analysis for Distributed Hard Real-Time Systems'. *Microprocessing and Microprogramming* **40**, 117–134.