# Optimal Utilization Bounds for the Fixed-priority Scheduling of Periodic Task Systems on Identical Multiprocessors*

Sanjoy K. Baruah

**Abstract**

In *fixed-priority* scheduling the priority of a job, once assigned, may not change. A new fixed-priority algorthm for scheduling systems of periodic tasks upon identical multiprocessors is proposed. This algorithm has an *achievable utilization* of $(m+1)/2$ upon $m$ unit-capacity processors. It is proven that this algorithm is optimal from the perspective of achievable utilization, in the sense that no fixed-priority algorithm for scheduling periodic task systems upon identical multiprocessors may have an achievable utilization greater than $(m+1)/2$.

**Keywords:** Real-time systems; Periodic task systems; Identical multiprocessors; Fixed-priority scheduling; Utilization bounds

## 1 Introduction

A **periodic task** $\tau_i = (C_i, T_i)$ is characterized by two parameters: an execution requirement $C_i$ and a minimum inter-arrival separation parameter $T_i$ (often referred to as the *period* of the task). A periodic task generates an infinite number of jobs, each having an execution requirement of $C_i$ and a deadline $T_i$ time units after its arrival time. The first job may arrive at any time-instant; successive arrivals are separated by at least $T_i$ time units. We use the notation $U(\tau_i)$ to denote the *utilization* of task $\tau_i$ — $U(\tau_i) \stackrel{\text{def}}{=} C_i/T_i$. A periodic task system consists of several such periodic tasks. Let $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ denote a periodic task system. For any such periodic task system $\tau$, $U_{\mathsf{sum}}(\tau)$ will denote the cumulative utilizations of all tasks in $\tau$ ($U_{\mathsf{sum}}(\tau) \stackrel{\text{def}}{=} \sum_{i=1}^{n} U(\tau_i)$), and $U_{\mathsf{max}}(\tau)$ will denote the largest utilization of any task in $\tau$ ($U_{\mathsf{max}}(\tau) \stackrel{\text{def}}{=} \max_{i=1}^{n} U(\tau_i)$). We will use the notation $I(\tau)$ to denote any collection of jobs generated by the periodic tasks in periodic task system $\tau$. We assume that each job is independent in the sense that it does not interact in any manner (accessing shared data, exchanging messages, etc.) with other jobs of the same or another task. We also assume that the model allows for job *preemption*; i.e., a job executing on a processor may be preempted prior to completing execution, and its execution may be resumed later, at no cost or penalty. (Note that what we call a periodic task here is sometimes referred to in the literature as a *sporadic* task.)

In this paper, we study the scheduling of periodic task systems on platforms that are comprised of $m$ ($\geq 1$) *identical multiprocessors*. We assume that interprocessor migration of jobs is permitted — i.e., a job that is executing upon a processor may be preempted and may later resume execution on a different processor. We do *not* permit job-level parallelism, i.e., a job executes on at most one processor at any instant of time. (In the remainder of this paper, we will assume that all processors have *unit* computing capacity; i.e., the amount of execution completed by executing a job for one unit of time is equal to one unit of execution. We will also assume that $U_{\mathsf{max}}(\tau) \leq 1$ for all task

*Supported in part by the National Science Foundation (Grant Nos. CCR-9988327, and ITR-0082866).

systems $\tau$, since a task system $\tau$ with $U_{\mathsf{max}}(\tau) > 1$ cannot be scheduled to meet all deadlines upon unit-capacity processors.)

***Run-time scheduling*** is the process of determining, during the execution of a real-time application system, which job[s] should be executed at each instant in time. Run-time scheduling algorithms for identical multiprocessor platforms are often categorized along two orthogonal axes: the *priority-assignment* axis and the *inter-processor migration* axis.

**Priority assignment.** Run-time scheduling algorithms are typically implemented as follows: at each time instant, assign a **priority** to each active job, and allocate the available processors to the highest-priority jobs. In **fixed-priority** scheduling, each job is assigned exactly *one* priority throughout its lifetime — the priority of a job, once assigned, cannot change. (We distinguish between such fixed-priority algorithms and *static priority* algorithms — static priority scheduling algorithms are fixed-priority algorithms with the additional constraint that all the jobs generated by each periodic task have the same priority. Thus, the *earliest deadline first* scheduling algorithm (EDF) [8, 2] is a fixed-priority scheduling algorithm but not a static-priority one; the *rate-monotonic* scheduling algorithm [8] is a static-priority algorithm for periodic tasks; and the *least-laxity* algorithm [11] is neither static- nor fixed-priority.)

It can be shown that the total number of processor preemptions (and interprocessor migrations, if permitted) in a schedule generated by any fixed-priority algorithm is bounded from above by the total number of jobs being scheduled. Hence, the preemption and migration costs in fixed-priority scheduled systems can be amortized across all the jobs in the system, by simply inflating the execution requirement of each job by the amount of work needed to perform one preemption and one inter-processor migration — this is indeed a very important advantage of fixed-priority scheduling schemes over schemes that are not fixed-priority.

**Interprocessor migration.** There have been two approaches towards scheduling of periodic tasks on multiprocessors: *partitioning* and *global scheduling*. In the partitioning approach, the tasks are statically partitioned among the processors, i.e., each task is assigned to a processor and is always executed on it. On the other hand, in global scheduling the tasks are not partitioned and all jobs are put in a single global queue. At each instant, the scheduler selects the $m$ highest priority jobs for execution, where $m$ is the number of processors. Under global scheduling, it is permitted that a job that has previously been preempted from one processor resume execution later upon a different processor.

**Utilization-based schedulability test.** The *schedulable utilization* of algorithms designed for scheduling periodic task systems is defined as follows:

**Definition 1 (Schedulable utilization)** "A scheduling algorithm can [correctly] schedule any set of periodic tasks [...] if the total utilization of the tasks is equal to or less than the schedulable utilization of the algorithm." [9, page 122]. ■

If the schedulable utilization of a scheduling algorithm is known, then a *sufficient* (albeit not necessary) test for determining whether a given periodic task system is correctly scheduled by this algorithm is easily designed: simply determine whether the utilization of the task system is equal to or less than the schedulable utilization of the algorithm. With respect to uniprocessor systems, it has been shown [8] that the schedulable utilization of EDF is one; and the schedulable utilization of

the rate-monotonic algorithm is $\ln 2 \approx 0.69$. (It has also been shown that these are optimal values, in the sense that the schedulable utilization of EDF cannot exceed one and that of the rate-monotonic algorithm cannot exceed $\ln 2$.) For multiprocessor scheduling using the partitioned approach, it has been shown that the schedulable utilization cannot exceed $(\frac{m+1}{2})$ upon $m$ processors; if the largest utilization $U_{\mathsf{max}}(\tau)$ of any task in $\tau$ is known, then a somewhat better bound of $(\frac{\beta m+1}{\beta+1})$ was proven by Lopez et al. [10], where $\beta = \lfloor 1/U_{\mathsf{max}}(\tau) \rfloor$.

**This paper.** In this paper, we propose a new fixed-priority scheduling algorithm to be used for the global scheduling of periodic task systems on multiprocessors. We prove that our algorithm has a schedulable utilization equal to $(m+1)/2)$ on $m$ identical processors — as $m \to \infty$, this bound approaches $m/2$ from above; hence, it follows that our algorithm successfully schedules any periodic task system with cumulative utilization at most $m/2$ on $m$ identical processors. Furthermore, we prove that no fixed-priority scheduling algorithm can have a schedulable utilization greater than $(m+1)/2$ upon $m$ identical processors; from the perspective of schedulable utilization, therefore, our scheduling algorithm is provably *optimal*.

**Organization.** The remainder of this paper is organized as follows. In Section 2, we briefly describe some prior results from the real-time scheduling literature that we need in order to prove the correctness of our proposed new algorithm. In Section 3, we present our new algorithm and prove several important properties, including its schedulable utilization and the fact that this schedulable utilization is the best possible. We conclude in Section 4 with a brief summary of the results presented here.

## 2 Background

In this section, we briefly review some results from multiprocessor real-time scheduling theory that we will need later in this paper. These results concern EDF *scheduling upon mutiprocessor platforms*, and the *predictability* of scheduling algorithms.

The Earliest Deadline First scheduling algorithm (EDF) is one of the most popular run-time scheduling algorithms. In EDF, jobs are assigned priorities in inverse proportion to their deadlines — the earlier the deadline, the higher the priority. EDF is known to be an *optimal* scheduling algorithm upon uniprocessors in the following sense: if any periodic task system can be correctly scheduled upon a given preemptive uniprocessor by *any* scheduling algorithm, then EDF will correctly schedule this task system on the given processor. Unfortunately, EDF is not optimal on multiprocessors in the same sense. There are nevertheless significant advantages to using EDFfor scheduling on multiprocessors if possible; consequently, the EDF-scheduling of periodic task systems upon identical multiprocessor platforms has recently attracted much attention (e.g., [10, 12, 4, 1]). The following theorem from [4] (which is independently derived, using different techniques, in [1]) will be used by us later in this paper.

**Theorem 1 ([4])** Periodic task system $\tau$ is scheduled to meet all deadlines by EDF on an identical multiprocessor platform comprised of $m$ unit-capacity processors, provided

$$U_{\mathsf{sum}}(\tau) \leq m - (m-1)\, U_{\mathsf{max}}(\tau) \tag{1}$$

■

Ha and Liu [6, 7, 5] have studied the issue of **predictability** in the multiprocessor scheduling of real-time systems from the following perspective.

**Definition 2 (Predictability)** Let us define a *job* $J_j = (r_j, e_j, d_j)$ as being characterized by an arrival time $r_j$, an execution requirement $e_j$, and a deadline $d_j$, with the interpretation that this job needs to execute for $e_j$ units over the interval $[r_j, d_j)$.

Let $A$ denote a scheduling algorithm, and $I = \{J_1, J_2, \ldots, J_n\}$ any set of $n$ jobs, $J_j = (r_j, e_j, d_j)$. Let $f_j$ denote the time at which job $J_j$ completes execution when $I$ is scheduled using some scheduling algorithm $A$.

Now, consider any set $I' = \{J'_1, J'_2, \ldots, J'_n\}$ of $n$ jobs obtained from $I$ as follows. Job $J'_j$ has an arrival time $r_j$, an execution requirement $e'_j \leq e_j$, and a deadline $d_j$ (i.e., job $J'_j$ has the same arrival time and deadline as $J_j$, and an execution requirement no larger than $J_j$'s). Let $f'_j$ denote the time at which job $J'_j$ completes execution when $I'$ is scheduled using algorithm $A$. Scheduling algorithm $A$ is said to be **predictable** if and only if for any set of jobs $I$ and for any such $I'$ obtained from $I$, it is the case that $f'_j \leq f_j$ for all $j$, $1 \leq j \leq n$.

■

Informally, Definition 2 recognizes the fact that the specified execution-requirement parameters of jobs are typically only *upper bounds* on the actual execution-requirements during run-time, rather than the exact values. For a predictable scheduling algorithm, one may determine an upper bound on the completion-times of jobs by analyzing the situation under the assumption that each job executes for an amount equal to the upper bound on its execution requirement; it is guaranteed that the actual completion time of jobs is no later than this determined value.

Since a periodic task system generates a set of jobs, Definition 2 may be extended in a straightforward manner to algorithms for scheduling periodic task systems. An algorithm for scheduling periodic task systems is predictable iff for any periodic task system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, the job completion time in the case when each job of $\tau_i$ has an execution requirement exactly equal to $C_i$ is an upper bound on the completion time of that job when every job of $\tau_i$ has an execution requirement of at most $C_i$, for all $i$, $1 \leq i \leq n$.

The result from the work of Ha and Liu [6, 7, 5] that we use can be stated as follows.

**Theorem 2 (Ha and Liu)** Any fixed-priority scheduling algorithm is predictable.

# 3 Algorithm fpEDF

In this section, we present Algorithm fpEDF, a fixed-priority scheduling algorithm for scheduling periodic task systems, and derive a utilization-based sufficient feasibility condition for it. In particular, we prove that any periodic task system with utilization at most $(m+1)/2$ can be scheduled by Algorithm fpEDF to meet all deadlines on $m$ unit-speed processors. Before presenting the algorithm, however, we need to prove a preliminary result (Theorem 3 below).

**Theorem 3** Any periodic task system $\tau$ satisfying the following two properties

***Property P1:*** $U_{\mathsf{sum}}(\tau) \leq \dfrac{m+1}{2}$

***Property P2:*** $U_{\mathsf{max}}(\tau) \leq \dfrac{1}{2}$

is correctly scheduled to meet all deadlines on $m$ processors by EDF.

**Proof:** By Theorem 1 (Condition 1), a sufficient condition for EDF-schedulability of $\tau$ is that

$$U_{\mathsf{sum}}(\tau) \leq m - (m-1)\, U_{\mathsf{max}}(\tau)$$

**Algorithm** fpEDF
Task system $\tau = \{\tau_1, \tau_2, \ldots \tau_n\}$ to be scheduled on $m$ processors
(It is assumed that $U(\tau_i) \geq U(\tau_{i+1})$ for all $i$, $1 \leq i < m$)

for $i = 1$ to $(m-1)$ do
  if $(U(\tau_i) > \frac{1}{2})$
    then $\tau_i$'s jobs are assigned highest priority (ties broken arbitrarily)
    else break
the remaining tasks' jobs are assigned priorities according to EDF

Figure 1: Algorithm fpEDF's priority-assignment rule

$$\begin{aligned}
&\equiv \left(\frac{m+1}{2} \leq m - (m-1)\,U_{\mathsf{max}}(\tau)\right) && \text{(By Property P1, above)}\\
&\equiv \left(\frac{m+1}{2} \leq m - (m-1)\cdot\frac{1}{2}\right) && \text{(By Property P2, above)}\\
&\equiv \left(\frac{m+1}{2} + \frac{m-1}{2} \leq m\right)\\
&\equiv (m \leq m)
\end{aligned}$$

and the theorem is proved. ■

### 3.1 Algorithm fpEDF: Description

We are now ready to describe Algorithm fpEDF, our fixed-priority scheduling algorithm for scheduling periodic task systems, and to derive a utilization-based sufficient feasibility condition for it.

Suppose that task system $\tau$ is to be scheduled by Algorithm fpEDF upon $m$ unit-capacity processors, and let $\{\tau_1, \tau_2, \ldots, \tau_n\}$ denote the tasks in $\tau$ indexed according to non-increasing utilization: $U(\tau_i) \geq U(\tau_{i+1})$ for all $i$, $1 \leq i < n$. Algorithm fpEDF first considers the $(m-1)$ "heaviest" (i.e., largest-utilization) tasks in $\tau$. All the tasks from among these heaviest $(m-1)$ tasks that have utilization greater than one-half are treated specially in the sense that all their jobs are always assigned highest priority (note that this is implemented trivially in an EDF scheduler by setting the deadline parameters of these jobs to $-\infty$). The remaining tasks' jobs — i.e., the jobs of the tasks from among the heaviest $(m-1)$ with utilization $\leq$ one-half, as well as of the $(n-m+1)$ remaining tasks — are assigned priorities according to their deadlines (as in "regular"EDF). This priority-assignment rule is presented in pseudo-code form, in Figure 1.

Note that Algorithm fpEDF reduces to "regular" EDF when scheduling $\tau$ upon $m$ processors if

1. $U_{\mathsf{max}}(\tau) \leq (1/2)$, in which case the "break" statement in the for-loop is executed for $i = 1$ and all tasks' jobs get EDF-priority; or
2. $m = 1$, in which case $(m-1) = 0$ and the for-loop is not executed at all.

**Computational complexity.** The runtime computational complexity of Algorithm fpEDF is identical to that of "regular" EDF, in the sense that, once it is determined which tasks' jobs always get highest priority, the runtime implementation of fpEDF is identical to that of EDF.

The process of determining which tasks' jobs always get highest priority would be performed according to the "for" loop in Figure 1 in time linear in $m$, if the tasks in $\tau$ are presented sorted according to utilization. If the tasks are not presorted according to utilization, then the $(m-1)$ heaviest tasks can be determined in time linear in $n$ using the standard median-find algorithm. Since $m \leq n$, in either case the computational complexity of the pre-runtime phase is thus $\mathcal{O}(n)$, where $n$ denotes the number of tasks in $\tau$.

### 3.2 Algorithm fpEDF: Properties

The following theorem states that Algorithm fpEDF correctly schedules on $m$ processors any periodic task system $\tau$ with utilization $U_{\mathsf{sum}}(\tau) \leq (m+1)/2$.

**Theorem 4** Algorithm fpEDF has a *schedulable utilization* (see Definition 1) of $\frac{m+1}{2}$ upon $m$ processors.

**Proof:** The proof is by induction on the number of processors $m$; we will prove (as a base case) that the theorem is true for $m = 1$, assume (as an inductive hypothesis) that it is true for $k-1$ processors and then prove that it must then be true for $k$ processors as well.

**Base ($m = 1$):** In this case, the utilization bound implied by the statement of the theorem is $(1+1)/2 = 1$; by the optimality of EDF on uniprocessors [8], we know that EDF can schedule any task system $\tau$ with $U_{\mathsf{sum}}(\tau) \leq 1$ upon a single processor.

**Inductive Hypothesis ($m = (k-1)$):** Assume that the statement of the theorem is true; i.e., Algorithm fpEDF correctly schedules upon $k-1$ processors any periodic task system with cumulative utilization at most $((k-1)+1)/2 \ = \ k/2$.

**Induction Step ($m = k$):** Consider now the case of $k$ processors. Consider any periodic task system $\tau$ satisfying $U_{\mathsf{sum}}(\tau) \leq (k+1)/2$. We consider two separate cases: **(i)** when $U_{\mathsf{max}}(\tau) \leq (1/2)$, and **(ii)** when $U_{\mathsf{max}}(\tau) > (1/2)$.

**(i)** If $U_{\mathsf{max}}(\tau) \leq 1/2$, then $\tau$ satisfies Properties P1 and P2 of Theorem 3. Therefore, it follows from Theorem 3 that $\tau$ is scheduled to meet all deadlines upon $k$ processors by EDF. Since Algorithm fpEDF reduces to EDF when no task has utilization $> (1/2)$, we conclude that Algorithm fpEDF correctly schedules $\tau$ upon $k$ processors.

**(ii)** Since $U_{\mathsf{max}}(\tau) > (1/2)$, Algorithm fpEDF assigns highest priority to all the jobs of $\tau_1$.

Consider the system $\tau'$ obtained from $\tau$ by removing the task $\tau_1 = (e_1, p_1)$ of maximum utilization:

$$\tau' \stackrel{\text{def}}{=} (\tau \setminus \{\tau_1\})$$

Observe that

$$\begin{aligned} U_{\mathsf{sum}}(\tau') &= U_{\mathsf{sum}}(\tau) - U(\tau_1) \\ &= U_{\mathsf{sum}}(\tau) - U_{\mathsf{max}}(\tau) \end{aligned}$$

$$\begin{aligned} & \leq \frac{k+1}{2} - \frac{1}{2} \\ \Rightarrow\ U_{\mathsf{sum}}(\tau') & \leq \frac{k}{2} \end{aligned}$$

By our inductive hypothesis above, Algorithm fpEDF therefore can successfully schedule $\tau'$ on $k-1$ processors.

Consider now the task system $\tau''$, comprised of $\tau'$ plus a task $\hat{\tau_1} = (p_1, p_1)$ with utilization 1 and period equal to the period of $\tau_1$:

$$\tau'' \stackrel{\text{def}}{=} (\tau \setminus \{\tau_1\}) \bigcup \{\hat{\tau_1} = (p_1, p_1)\}$$

A schedule for $\tau''$ on $k$ processors can be obtained from the fpEDF schedule for $\tau'$ on $k-1$ processors (which, according to our inductive hypothesis, is guaranteed to exist), by simply devoting one processor exclusively to the additional task $\hat{\tau_1}$, and scheduling the remaining $(k-1)$ exactly as in the fpEDF-schedule.

Furthermore, this schedule is exactly equivalent to the one that would be generated if Algorithm fpEDF were scheduling $\tau''$ on $k$ processors — this follows from the observations that

- since task $\hat{\tau_1}$ has the highest utiliation of any task in $\tau''$, its jobs would be assigned highest priority by Algorithm fpEDF;
- jobs of the remaining tasks in $\tau''$ would be assigned exactly the same priorities as they would in the $(k-1)$-processor fpEDF-schedule of $\tau'$; and
- the jobs of $\hat{\tau_1}$ completely occupy one processor (since $U(\tau_1) = 1$).

Thus, Algorithm fpEDF successfully schedules task-system $\tau''$ upon $k$ processors. Since Algorithm fpEDF is a fixed-priority algorithm, it follows by Theorem 2 that it is *predictable*; by the definition of predictability, it follows that Algorithm fpEDF successfully schedules $\tau$, since $\tau$ may be obtained from $\tau''$ by reducing the execution requirement of each of $\hat{\tau_1}$'s jobs by a quantity $(p_1 - e_1)$.

■

We show below (Theorem 5) that no scheduling algorithm that belongs to the family of algorithms to which EDF and fpEDF belong — *fixed-priority* scheduling algorithms — can have a greater schedulable utilization than Algorithm fpEDF.

Recall that a *fixed-priority* scheduling algorithm 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 instant in time, then $J_i$ always has higher priority than $J_j$. In other words, individual jobs are assigned fixed priorities (although different jobs of the same task may have very different priorities).

**Theorem 5** No $m$-processor fixed-priority scheduling algorithm has a schedulable utilization greater than $\frac{m+1}{2}$.

**Proof:** Consider the periodic task system comprised of $m+1$ identical tasks, each with execution requirement $1+\epsilon$ and period 2, where $\epsilon$ is an arbitrarily small positive number. Each task releases its first job at time-instant zero. Any fixed-priority schedule must assign these jobs fixed priorities relative to each other and the task whose job is assigned the lowest priority at time-instant zero misses its deadline. Note that as $\epsilon \to 0$, $U_{\mathsf{sum}}(\tau) \to \frac{m+1}{2}$; thus, the required result follows. ■

As with partitioned scheduling [10], we can obtain better bounds upon schedulable utilization if the largest utilization $U_{\mathsf{max}}(\tau)$ of any task in $\tau$ is known:

**Theorem 6** Algorithm fpEDF correctly schedules any periodic task system $\tau$ satisfying

$$U_{\mathsf{sum}}(\tau) \leq \max\left(m - (m-1)U_{\mathsf{max}}(\tau), \frac{m}{2} + U_{\mathsf{max}}(\tau)\right) \tag{2}$$

upon $m$ unit-capacity processors. (The bound of Equation 2 is depicted graphically in Figure 2.)

**Proof:** We consider two cases separately: **(i)** when $U_{\mathsf{max}}(\tau) \leq (1/2)$, and **(ii)** when $U_{\mathsf{max}}(\tau) > (1/2)$. For $U_{\mathsf{max}}(\tau) \leq (1/2)$, observe that the first term in the "max" above is $\geq$ the second, while for $U_{\mathsf{max}}(\tau) < (1/2)$, the second term in the "max" is $\geq$ the first.

**(i):** For $U_{\mathsf{max}}(\tau) \leq (1/2)$, it is the first term in the "max" that defines the schedulable utilization for task systems $\tau$ satisfying $U_{\mathsf{max}}(\tau) \leq 1/2$. That is,

$$U_{\mathsf{sum}}(\tau) \leq m - (m-1)U_{\mathsf{max}}(\tau) \tag{3}$$

for such systems.

However we have already observed in Section 3.2 that Algorithm fpEDF behaves exactly as EDF does when $U_{\mathsf{max}}(\tau) \leq 1/2$. The correctness of the theorem follows from the observation that the bound of Equation 3 above is the EDF-bound of Theorem 1 (Equation 1).

**(ii):** As in the proof of Theorem 4, let $\tau'$ denote the task system obtained from $\tau$ by removing the task $\tau_1$ of maximum utilization:

$$\tau' \stackrel{\text{def}}{=} (\tau \setminus \{\tau_1\})$$

As explained in the proof of Theorem 4, a sufficient condition for $\tau$ to be correctly scheduled on $m$ processors by Algorithm fpEDF is that $\tau'$ be correctly scheduled on $(m-1)$ processors by Algorithm fpEDF. That is

$$\begin{aligned}
&\tau \text{ is correctly scheduled on } m \text{ processors} \\
&\quad \Leftarrow \quad \tau' \text{ is correctly scheduled on } m-1 \text{ processors} \\
&\quad \Leftarrow \quad U_{\mathsf{sum}}(\tau') \leq \frac{(m-1)+1}{2} \\
&\quad \equiv \quad (U_{\mathsf{sum}}(\tau) - U_{\mathsf{max}}(\tau)) \leq \frac{m}{2} \\
&\quad \equiv \quad U_{\mathsf{sum}}(\tau) \leq U_{\mathsf{max}}(\tau) + \frac{m}{2}
\end{aligned}$$

which is as stated in the theorem.

■

# 4 Conclusions

We have presented a new a fixed-priority scheduling algorithm for scheduling periodic task systems upon identical multiprocessors. We have proved that our algorithm is optimal from the perspective of schedulable utilization in the sense that it successfully schedules any periodic task system with utilization $\leq (m+1)/2$ upon $m$ identical processors, and no fixed-priority scheduling algorithm can have a greater schedulable utilization than our algorithm does. We have also determined the schedulable utilization as a function of the maximum utilization of the task system being scheduled.

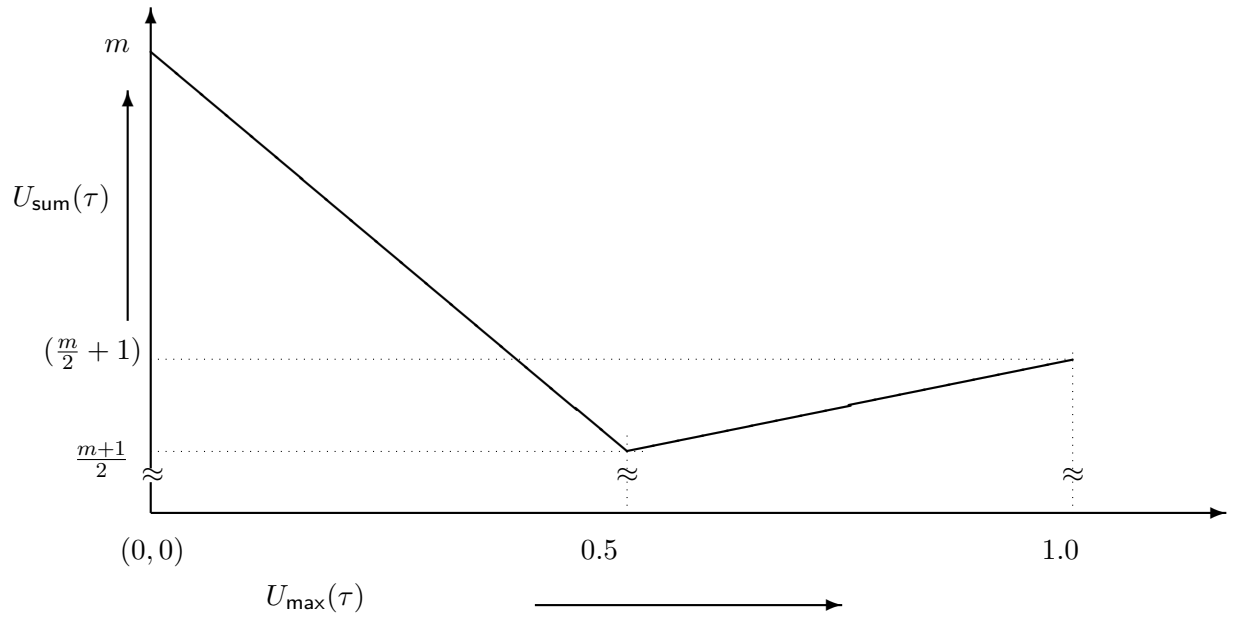


Figure 2: **Utilization bounds for an $m$-processor system, with each processor having unit computing capacity. Utilization bounds are plotted on the $y$-axis, and the maximum utilization on the $x$-axis.**

# References


[1] Baker, T. P. An analysis of EDF schedulability on a multiprocessor. Tech. Rep. TR-030202, Department of Computer Science, Florida State University, 2003.

[2] Dertouzos, M. Control robotics : the procedural control of physical processors. In *Proceedings of the IFIP Congress* (1974), pp. 807–813.

[3] Dhall, S. K., and Liu, C. L. On a real-time scheduling problem. *Operations Research 26* (1978), 127–140.

[4] Goossens, J., Funk, S., and Baruah, S. Priority-driven scheduling of periodic task systems on multiprocessors. *Real Time Systems*. To appear.

[5] Ha, R. *Validating timing constraints in multiprocessor and distributed systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1995. Available as Technical Report No. UIUCDCS-R-95-1907.

[6] Ha, R., and Liu, J. W. S. Validating timing constraints in multiprocessor and distributed real-time systems. Tech. Rep. UIUCDCS-R-93-1833, Department of Computer Science, University of Illinois at Urbana-Champaign, October 1993.

[7] Ha, R., and Liu, J. W. S. Validating timing constraints in multiprocessor and distributed real-time systems. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems* (Los Alamitos, June 1994), IEEE Computer Society Press.

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

[9] Liu, J. W. S. *Real-Time Systems*. Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458, 2000.

[10] Lopez, J. M., Garcia, M., Diaz, J. L., and Garcia, D. F. Worst-case utilization bound for EDF scheduling in real-time multiprocessor systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems* (Stockholm, Sweden, June 2000), IEEE Computer Society Press, pp. 25–34.

[11] Mok, A. K. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.

[12] Srinivasan, A., and Baruah, S. Deadline-based scheduling of periodic task systems on multiprocessors. *Information Processing Letters 84*, 2 (2002), 93–98.