Liu and Layland’s schedulability test revisited

Raymond Devillers*  Joël Goossens*

December 28, 1999

Abstract

In this paper, we study the problem of scheduling hard real-time periodic task sets on a single processor with the rate monotonic scheduler. We are concerned with the feasibility test given by Liu and Layland, based on the least upper bound of the utilization factor. We show that the result is incomplete and that the argument is incorrect. We complete and correct the result.

Key words: real-time systems, periodic task sets, synchronous systems, rate monotonic scheduling, feasibility test.

1 Introduction

We shall consider in this paper the problem of scheduling a set of $n$ periodic independent real-time tasks $\tau_1, \ldots, \tau_n$ on a single (preemptible) processor with static priorities, i.e., the successive requests of a same task all have the same fixed priority. Each periodic task $\tau_i$ is characterized by the pair $(T_i, C_i)$ with $0 < C_i \leq T_i$, i.e., by a period $T_i$, and an execution time $C_i$. Moreover, we consider synchronous systems, where all tasks start their execution at time 0, and we suppose that the system itself starts at time 0. The execution of the $k^{th}$ request of task $\tau_i$, which occurs at time $(k - 1) \cdot T_i$, must finish before or at time $k \cdot T_i$,

*Université Libre de Bruxelles, Département d’Informatique CP 212, Boulevard du Triomphe, 1050 Brux- seils, Belgium.
i.e., before the next request of the same task; a deadline failure is fatal for the system: the
deadlines are considered to be hard. We shall consider the rate monotonic scheduler \[3\],
RMS for short, which is optimal among\footnote{It may be noticed that for other deadline constraints, or if dynamic strategies are allowed (e.g., the
deadline driven scheduler for which the successive requests of a task have different priorities, or the least
laxity first scheduler for which the priority of each request changes with time), or if the various tasks do
not start at the same time, the rate monotonic scheduler is not optimal, but we consider here synchronous
systems where the processing of each request must simply finish before the next request of the same task,
and we consider only static priority strategies.} all the static and preemptive schedulers for the
kind of synchronous and late deadline systems considered here: it assigns a priority to each
task \(\tau_i\) in proportion of its arrival rate \(\frac{1}{T_i}\) (if many tasks have the same period, they may
be combined in a single one with the same period and the accumulated execution time, so
that the tie may be broken in an arbitrary way). All timing characteristics of the tasks in
our model of computation are assumed to be natural integers. In addition we shall assume
that the switching times (including scheduling) may be neglected.

The rest of the paper is organized thus: we shall first show why the reasoning of Liu and
Layland is incorrect; we shall then correct the result. Finally we shall complete the result
by showing explicitly that the condition provides a sufficient test for the feasibility of the
system.

2 The utilization factor based feasibility test

We shall now consider the \textit{feasibility problem}, i.e., deciding if all the deadlines will be res-
spected. Liu and Layland have defined a rather efficient sufficient condition for the schedu-
lability of a task set, based on the \textit{utilization factor}, i.e., the (long term) fraction of the
processor time spent in the execution of the task set (if feasible):

\[ U = \sum_{i=1}^{n} \frac{C_i}{T_i} \quad (U \in \mathbb{Q}). \]

Liu and Layland have shown that if the utilization factor is less than 60% the set is schedu-
lable with the rate monotonic rule. However while the end result is correct, the reasoning
of Liu and Layland, which is regularly reproduced in the literature \[4, 1\], to justify this
feasibility test is incorrect and incomplete; we shall present the completed and corrected result here.

**Definition 1** A task set $\tau_1, \ldots, \tau_n$ is said to fully utilize the processor if the task set is schedulable and if an increase of any $C_i$ ($1 \leq i \leq n$) makes the task set unschedulable. \[\square\]

A well known characterization of task sets (of the kind we consider here) which fully utilize the processor under RMS is the following:

**Theorem 2** A synchronous task set $\tau_1, \ldots, \tau_n$ scheduled with RMS fully utilizes the processor if and only if the subset $\tau_1, \ldots, \tau_{n-1}$ is schedulable and $C_n$ is the number of processor time units left free by the other tasks up to $T_n$. \[\square\]

This itself immediately results from the well known (see for instance [3, 2]) property:

**Theorem 3** For a synchronous system with late deadlines and static priorities, the first request of each task has the worst response time among all requests of that task. \[\square\]

**Definition 4** We define $b_n$, the least upper bound of the utilization factor over for $n$ tasks, as the lower bound of $U$ among all the sets of $n$ tasks which fully utilize the processor. \[\square\]

The sufficient feasibility test given by Liu and Layland is based on the computation of the bound $b_n$. They have stated that $b_n = n(\sqrt{2} - 1)$, but this property was not really proved in their paper: the result is correct but their argument is not satisfactory. We shall also prove at the end of the paper that those bounds $b_n$ give a sufficient feasibility test for RMS-scheduled tasks sets.

We give here the first part of the original proof of Liu and Layland, which is first given while assuming that the ratio between any task periods is less than 2 (and consequently $T_n < 2T_1$ assuming that $T_n > T_{n-1} > \cdots > T_1$), and we shall exhibit why their argument is incorrect.

**Proof from Liu and Layland [3]**. Let $\tau_1, \ldots, \tau_n$ denote the $n$ tasks. Let $C_1, \ldots, C_n$ be the execution times of the tasks that fully utilize the processor and minimize the processor utilization factor (here we shall allow real-valued $C_i$'s, considering they may be arbitrarily
approximated by rational values, hence by integer values up to a multiplication of all task characteristics by an adequate integer coefficient. We assume that $T_n > T_{n-1} > \cdots > T_1$ (two tasks with the same period can be represented by a single task with the same period and a computation time equal to the sum of the original computation times) and $T_n < 2T_1$. In this situation we must have: $C_i = T_{i+1} - T_i$ ($i < n$) and $C_n = T_n - 2(C_1 + \cdots + C_{n-1})$; consider first the case of $C_1$:

1. If $C_1 = T_2 - T_1 + \Delta$ ($\Delta > 0$), the task set $\tau_1', \ldots, \tau_n'$ with $T_i' = T_i \forall i$ and $C_i' = T_2 - T_1$, $C_2' = C_2 + \Delta$, $C_3' = C_3$, ..., $C_n' = C_n$ also fully utilizes the processor (see Figure 1, left); the feasibility of the system and the full utilization of the processor result from the fact that nothing changes for $\tau_3, \ldots, \tau_n$ up to $T_n$ - which occurs before $2T_1$ - from the fact that the first two requests of $\tau_1$ and $\tau_2$ simply exchange a $\Delta$-slot, and from theorems 2, 3. And $U - U' = \frac{\Delta}{T_1} - \frac{\Delta}{T_2} > 0$ (contradicting our hypothesis that the utilization factor of $\tau_1, \ldots, \tau_n$ is minimal).

2. If $C_1 = T_2 - T_1 - \Delta$ ($\Delta > 0$), the task set $\tau_1'', \ldots, \tau_n''$ with $T_i'' = T_i \forall i$ and $C_i'' = C_1 + \Delta$, $C_2'' = C_2 - 2\Delta$, $C_i'' = C_i$ for $i = 3, 4, \ldots, n$, also fully utilizes the processor (see Figure 1, right), with $U - U'' = -\frac{\Delta}{T_1} + \frac{2\Delta}{T_2} > 0$ (contradicting our hypothesis that the utilization factor of $\tau_1, \ldots, \tau_n$ is minimal).

\[\]

Figure 1: Schedule of task set \{\tau_1, \tau_2\}, if $C_1 = T_2 - T_1 + \Delta$ (left) and $C_1 = T_2 - T_1 - \Delta$ (right).

The reasoning of Liu and Layland is not valid, however: the second part of their proof (the

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]
The task set fully utilizes the CPU.

The task set does not fully utilize the CPU: the system is idle in the interval [13, 15].

The case where \( C_1 = T_2 - T_1 - \Delta (\Delta > 0) \) is incorrect for at least two reasons:

- The proof assumes that \( C_2 > 2\Delta \), without justification.
- Even so, the task set \( \tau'_1, \ldots, \tau'_n \) does not necessarily fully utilize the CPU, as exhibited by the following example.

**Example 5** Consider the task set \( \{T_1 = 8, C_1 = 2\}, \{T_2 = 11, C_2 = 3\}, \{T_3 = 15, C_3 = 5\} \)

which fully utilizes the CPU as exhibited in Figure 2, with \( U = \frac{10}{33} \). If we choose \( C'_1 = T_2 - T_1 = C_1 + \Delta = 3, C'_2 = C_2 - 2\Delta = 1 \) and \( C'_3 = C_3 \), the set does not fully utilize the CPU as exhibited in Figure 3. And if we try to correct the situation by increasing \( C_3 \), i.e., if we choose \( C''_3 = C_3 + 2\Delta \), the set fully utilizes the CPU but in this case \( U'' = \frac{22}{33} > U \).

This shows that the argument of Liu and Layland is wrong and may not be used to exhibit a contradiction on the assumed minimality of the considered utilization factor.

It may be noticed that Serlin [5] has "shown" this property with a slightly different technique.
but Serlin assumes \( C_i = T_{i+1} - T_i \) \( (i < n) \) without any proof, hence the proof of Serlin is also incomplete.

Remark however that Example 5 does not contradict the fact that we must have \( C_i = T_{i+1} - T_i \) \( (i < n) \) and \( C_n = T_n - 2(C_1 + \cdots + C_{n-1}) \), since in the previous example if we choose \( C''_1 = 3, C''_2 = 4 \) and \( C''_3 = 1 \), the set fully utilizes the CPU and \( U'' = \frac{103}{132} < U \).

Indeed, while their argument was wrong, Liu and Layland had the correct intuition. To prove it, one simply has to reorder the subgoals.

**Lemma 6** For a set of \( n \) periodic tasks and the restriction that the ratio between any two task periods is less than 2, \( b_n = n(\sqrt[3]{2} - 1) \), for the rate monotonic priority rule.

**Proof.** We shall first show that, in the computation of \( b_n \), we may restrict our attention to systems of \( n \) tasks \( \tau_1, \ldots, \tau_n \) (with \( T_n > T_{n-1} > \cdots > T_1 \) and a full utilization of the processor) such that \( \forall i < n : C_i \leq T_{i+1} - T_i \). The argument is similar to the one used by Liu and Layland.

Consider first the case of \( C_1 \) and suppose that \( C_1 = T_2 - T_1 + \Delta \) \( (\Delta > 0) \); notice that we must have that \( T_2 < 2T_1 \), otherwise \( C_1 > T_1 \) and the task set may not be schedulable; now, the task set \( \tau'_1, \ldots, \tau'_n \) with \( T'_i = T_i \) \( \forall i \) and \( C'_1 = T_2 - T_1 \), \( C'_2 = C_2 + \Delta \), \( C'_3 = C_3 \), ..., \( C'_n = C_n \) also fully utilizes the processor (see Figure 1, left, with the same comments as above); \( U' - U'' = \frac{1}{T_1} - \frac{1}{T_2} > 0 \), since \( T_2 > T_1 \), hence systems with \( C_1 > T_2 - T_1 \) do not afford anything to the infimum of \( U \).

We can apply the same argument for \( C_2, \ldots, C_{n-1} \).

Next, we observe that, in the computation of \( b_n \), we may restrict our attention to systems fully utilizing the processor such that \( \forall i < n : C_i = T_{i+1} - T_i \). We already know from the previous point that we may assume that \( \forall i < n : C_i \leq T_{i+1} - T_i \); then, since we have to fully utilize the processor, it occurs from Theorem 2 and the fact that each task \( \tau_i \) with \( i < n \) exactly issues and completes 2 requests before \( T_n \) (see Figure 4) that \( C_n = T_n - 2\sum_{i=1}^{n-1} C_i \) (the first \( n-1 \) tasks are schedulable and between 0 and \( T_1 \), as well as between \( T_1 \) and \( T_n \)), they use \( \sum_{i=1}^{n-1} C_i \) time units, with \( \sum_{i=1}^{n-1} C_i \leq T_n - T_1 < T_1 \).

Now, if \( C_1 = T_2 - T_1 - \Delta \) \( (\Delta > 0) \), from the previous observations, the task set \( \tau''_1, \ldots, \tau''_n \)
with $T_i'' = T_i \forall i$ and $C_i' = C_1 + \Delta = T_2 - T_1$, $C_i'' = C_n - 2\Delta$, $C_i'' = C_i$ for $i = 2, 3, \ldots, n - 1$, also fully utilizes the processor, and $U' - U'' = -\frac{\Delta}{T_1} + \frac{2\Delta}{T_n} > 0$ since $2T_1 > T_n$; hence systems with $C_1 < T_2 - T_1$ do not afford anything to the infimum of $U$. We can apply the same argument for $C_2, \ldots, C_{n-1}$ and we get our property.

![Diagram](image)

Figure 4: System with $C_i \leq T_{i+1} - T_i \ \forall i = 1, \ldots, n - 1$.

The rest of the proof is as in [3]: let $g_i = \frac{T_n - T_i}{T_i} \ (i = 1, \ldots, n - 1)$; we get

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = 1 + g_1 \frac{g_1 - 1}{g_1 + 1} + \sum_{i=2}^{n-1} g_i \frac{g_i - g_{i-1}}{g_i + 1}$$

This expression must be minimal, hence $\frac{dU}{dg_j} = \frac{2g_j - 2g_{j+1} + g_j^{2} - g_{j+1}^{2}}{g_j + 1} = 0$, for $j = 1, \ldots, n - 1$.

The general solution can be shown to be $g_j = 2 \frac{n-j}{n} - 1 \ (j = 1, \ldots, n - 1)$.

It follows that $b_n = n(\sqrt{n} - 1)$.

It may be noticed that $b_n < b_{n-1}$ and that $\lim_{n \to \infty} b_n = \ln 2$ (by the l'Hospital rule).

Moreover, except for the trivial case $n = 1$, the bound $b_n$ is never reached, since it is irrational, while from our assumptions $U$ is always rational.

The restriction that the ratio between task periods is less than 2 can now be relaxed (in this case, the argument of Liu and Layland [3] was perfectly valid).

**Theorem 7 ([3])** For a set of $n$ periodic tasks, $b_n = n(\sqrt{n} - 1)$, for the rate monotonic scheduler.
Liu and Layland [3] have formulated a sufficient condition for the schedulability of a task set based on the bound $b_n$, without a proof. The authors, from Theorem 7, stated that: if a set of $n$ tasks has an utilization factor less than the upper bound $b_n$, it follows that the set is schedulable. This property does not immediately follow from Theorem 7, however, since the least upper bound $b_n$ concerns the utilization factor of schedulable task sets only. There is no a priori reason to think that there are no unschedulable task set with a utilization factor less than $b_n$. The property is true however, but it does not follow only from Theorem 7 and Definition 4, but also from the fact that $b_n$ is strictly decreasing in $n$.

**Theorem 8** Let $\tau_1, \ldots, \tau_n$ be a task set. If $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq b_n$, then the task set is schedulable.

**Proof.** By induction on $n$. The property is trivially true for $n = 1$: $\tau_1$ is schedulable iff $\frac{C_1}{T_1} \leq 1 = b_1$. Let us assume that the property is true up to $n - 1$, and consider a set of $n$ tasks $\tau_1, \ldots, \tau_n$ with $U = \sum_{i=1}^{n} \frac{C_i}{T_i} < b_n$. Since $b_n < b_{n-1}$, we have that $U_n < b_{n-1}$. Consider the $n-1$ highest priority tasks $\tau_1, \ldots, \tau_{n-1}$ with $U_{n-1} = \sum_{i=1}^{n-1} \frac{C_i}{T_i} = U_n - \frac{C_n}{T_n} < U_n$, hence $U_{n-1} < b_{n-1}$ and by induction hypothesis the task subset $\tau_1, \ldots, \tau_{n-1}$ is schedulable.

Suppose now that $\tau_1, \ldots, \tau_n$ is not schedulable; since the first $n-1$ tasks are schedulable, we have: $\exists x : 0 \leq x < C_n$ : with $C'_1 = C_1, C'_2 = C_2, \ldots, C'_{n-1} = C_{n-1}$, and $C'_n = x$ so that the task set is schedulable, while with $C'_1 = C_1, C'_2 = C_2, \ldots, C'_{n-1} = C_{n-1}$, and $C'_n = x + 1$ the task set is not schedulable; hence the task set with $C'_n = x$ fully utilizes the processor and $U'_n = \sum_{i=1}^{n-1} \frac{C'_i}{T_i} + \frac{x}{T_n} < U_n < b_n$. By definition of $U_n$ and $U'_n$ this leads to a contradiction and proves the theorem. \[\square\]

This theorem gives us a sufficient condition. The series $b_n$ converges to $\ln 2$. Hence, we are always sure that any task set (for any $n$) with a utilization factor less than 0.69 is schedulable, since 0.69 < $\ln 2 < b_n < b_{n-1} < \cdots < b_1$. 


3 Conclusion

In this paper we have studied the feasibility problem of periodic task sets. We have considered the feasibility test based on the least upper bound of the utilization factor. We have shown that the argument used in the literature is incorrect and incomplete. We have corrected and completed the result.

References


