Sufficient Schedulability Tests for EDF-Scheduled Real-Time Systems under Interference of a High Priority Task

J. Augusto Santos-Jr. and George Lima

Distributed Systems Laboratory (LaSiD)
Computer Science Department
Federal University of Bahia (UFBA)
Brazil, Salvador-Bahia
{jamjunior,gmlima}@ufba.br

Abstract

We provide four new schedulability tests for a scheduling model according to which there is a high priority task concurrently executed with a set of EDF-scheduled tasks. All tests are proved correct and have their performance evaluated by simulation.

1. Introduction

In this paper we address the schedulability analysis problem. More specifically, we are interested in verifying whether a given uniprocessor real-time system, composed of a set of sporadic, independent tasks with implicit deadlines, equipped with a scheduling algorithm, meet all deadlines. In terms of schedulability analysis, classical results for the assumed task model are known [1]. For example, if tasks are scheduled by the Earliest Deadline First (EDF) algorithm, all deadlines are met if and only if the task set does not require more than 100% of processing resources. When the Rate Monotonic (RM) algorithm is considered, no deadline is missed provided that no more than \( n(2^{\frac{1}{n}} - 1) \) of the processor is used. Although using EDF provides higher system utilization, there are some situations when it is advantageous to use fixed-priority scheduling algorithms such as RM.

For example, interrupt and exception handlers or other checking routines, called here urgent routines, are usually short piece of codes that need to be executed at high frequency in real-time operating systems. It is recommended that they execute at the highest priority level so as to minimize internal latencies. Those requirements fit well in the fixed-priority scheduling model, where the highest priority is assigned to urgent routines. Doing so, however, reduces the achievable system utilization. On the other hand, using a dynamic-priority scheduling algorithm such as EDF does not offer guarantees that urgent routines are always executed at with highest priority making them subject to preemption and execution delays.

In this paper we provide the means of analyzing EDF-scheduled systems that implements urgent routines as the highest priority task. More specifically, the assumed scheduling model has two fixed priorities. The highest priority is reserved to execute urgent routines whereas all other system tasks are scheduled by EDF within the lowest priority level. Fig. 1 illustrates this scheme showing the schedule for three periodically activated tasks. Task activation instants are indicated by vertical arrows and their execution is represented by white boxes. As can be noted, all tasks finish their execution before its next activation time, considered here as deadlines. If this task set were scheduled in a purely fixed-priority fashion, the lowest priority task would miss its deadline.

![Figure 1. EDF-scheduled tasks suffering interference due to the execution of a high priority task.](image-url)
analysis problem considering the model described in
Fig. 1. They developed an exact schedulability test for
this model. That is, their test is capable of precisely
identifying both schedulable and non-schedulable sys-
tems. Later on, Gonzalez and Palencia [3] have also
presented an exact test for a more general framework
according to which the scheduler handles several prior-
ity levels. Within each level tasks can be scheduled
either in a fixed-priority fashion or by EDF. However,
both tests run in exponential time, as it is common for
exact tests in EDF-scheduled systems [4], [5].

Unlike these schemes, we are interested in schedu-
ability tests with low computational complexity for
the model depicted in Fig. 1. Four such tests are
derived. They provide sufficient schedulability condi-
tions, which means that all non-schedulable systems
are identified but some schedulable systems may be
not. However, experiments carried out on synthetic
generated systems indicate that the proposed tests
perform very well for systems that use up to 95% of
processor. We also show in this paper that checking
schedulability for the model described in Fig. 1 can
be done via well known schedulability tests for fixed-
priority systems [1], [6], [7] applied to two tasks, one
being the urgent routines and the other representing all
EDF-scheduled tasks.

After introducing the computation model and nota-
tion in Section 2, the proposed schedulability tests are
described in Section 3 and evaluated by simulation in
Section 4. Out final comments are given in Section 5.

2. Notation and System Model

We consider a set of tasks \( \Gamma = \{ \tau_1, \tau_2, \ldots, \tau_n \} \)
to be scheduled on a single processor according to
EDF and assume that there is a high priority task,
\( \tau_0 \), that may interfere in the execution of any task
in \( \Gamma \). Task \( \tau_0 \) represents urgent routines that require
minimum latencies. Examples of urgent routines are
error checking, interrupt handlers, etc. Tasks in \( \Gamma \)
are assumed to be fully preemptable and all tasks are
independent of one another.

Any task \( \tau_i \) in the system is denoted by a tuple
\((C_i, T_i)\), where \( C_i \leq T_i \) represents its maximum
required computation time and \( T_i \) is its minimum inter-
arrival time, also called period. We denote \( U(\tau_i) = \frac{C_i}{T_i} \)
the utilization of a task \( \tau_i \) and the utilization of a
task set \( \Gamma \) is denoted \( U(\Gamma) = \sum_{\tau_i \in \Gamma} U(\tau_i) \). If a task
\( \tau_i \) arrives at time \( t \), the system must schedule it for
execution so that \( C_i \) processor units are allocated to
\( \tau_i \) within \([t, t + T_i]\). That is, we assume an implicit-
deadline task model.

We also assume that the period of \( \tau_0 \) is not greater
than the period of any other task in \( \Gamma \). This assumption
is not necessary for all derived tests and it mostly
comes from the optimality of the Rate-Monotonic pri-
ority assignment [1]. As \( \tau_0 \) represents urgent routines, which usually require a low activation period, this
assumption does not restrict the applicability of results
presented in this paper.

3. Schedulability tests

We now start deriving the new sufficient schedu-
lability tests. Theorems 1, 2, 3 show three of them
by establishing a bound on processor utilization above
which the system is considered not schedulable. The
fourth test explores the result stated in Theorem 4
and use response-time analysis to derive schedulability
conditions for the set \( \Gamma \).

Theorem 1 (Test 1): Let \( \Gamma = \{ \tau_1, \tau_2, \ldots, \tau_n \} \) be a
set of tasks scheduled by EDF and let \( \tau_0 \) be the highest
priority task. There is no deadline miss provided that
Eq. (1) holds.

\[
\left( \frac{T_0}{\min_{\tau_i \in \Gamma}(T_i)} + 1 \right) U(\tau_0) + U(\Gamma) \leq 1 \tag{1}
\]

Proof: From the assumed model, \( \tau_0 \) does not miss
its deadline and so assume that some task \( \tau_i \in \Gamma \)
misses its deadline at some time \( d \). Let \( r < d \) be its
release time. Also, consider the last time \( t < d \) so
that the processor is not idle within \([t, d]\) but is idle
just before \( t \). If such a time does not exist, let \( t = 0 \).
Note that \( t \leq r \) and \( d - r = T_i \). To simplify notation,
let \( \Delta = r - t \). Let us compute the maximum demand
within \([t, d]\) from those tasks that may interfere in the
execution of \( \tau_i \). As for tasks in \( \Gamma \), we must account
for the execution of tasks whose jobs have deadlines
less than or equal to \( d \). Also, since \( \tau_0 \) interferes in
the execution of any task in \( \Gamma \), its activation within
\([t, d]\) must be accounted for. It is known that \( \tau_0 \)
does not arrive more than \( \frac{T_i + \Delta}{T_0} \) times during \([t, d]\).
Computing the total demand and considering that \( \tau_i \)
misses its deadline yields

\[
\left[ \frac{T_i + \Delta}{T_0} \right] C_0 + \sum_{\tau_i \in \Gamma} \left[ \frac{T_i + \Delta}{T_j} \right] C_j > T_i + \Delta
\]

\[
\left( \frac{T_i + \Delta}{T_0} + 1 \right) C_0 + U(\Gamma)(T_i + \Delta) > T_i + \Delta
\]

\[
U(\tau_0) + \frac{U(\tau_0)T_0}{T_i + \Delta} + U(\Gamma) > 1
\]

\[
\left( \frac{T_0}{T_i} + 1 \right) U(\tau_0) + U(\Gamma) > 1 \tag{2}
\]
Condition (2) must hold for any task \( \tau_i \in \Gamma \) that misses its deadline. This implies that Eq. (1) is a sufficient schedulability test, as required.

The second schedulability test takes advantage of periods that are multiple of the period of the highest priority task. It is required that \( T_0 \leq \min_{\tau_i \in \Gamma}(T_i) \).

**Theorem 2 (Test 2):** Let \( \Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\} \) be a set of tasks scheduled by EDF and let \( \tau_0 \) be the highest priority task in the system such that \( T_0 \leq \min_{\tau_i \in \Gamma}(T_i) \). There is no deadline miss provided that Eq. (3) holds.

\[
U(\tau_0) + \sum_{\tau_i \in \Gamma} \frac{T_i}{T_0} U(\tau_i) \leq 1 \tag{3}
\]

**Proof:** As \( \tau_0 \) cannot miss its deadline, let us focus on tasks in \( \Gamma \). Consider a task set \( \Gamma' \) obtained from \( \Gamma \) as follows. For each task \( \tau_i = (C_i, T_i) \in \Gamma \) there is a task \( \tau'_i = (C'_i, T_0) \) in \( \Gamma' \), where

\[
C'_i = \frac{C_i}{T_0}, 
\]

and so \( U(\Gamma') = \sum_{\tau_i \in \Gamma'} \frac{T_i}{T_0} U(\tau_i) \).

Now consider scheduling \( \tau_0 \) and \( \Gamma' \) with \( \tau_0 \) being the highest priority task and \( \Gamma' \) being scheduled by EDF. The worst-case response time of any task \( \tau'_i \) is equal to \( C + \sum_{\tau_j \in \Gamma'} C'_j \). This means that \( \tau'_i \) meets its deadline if \( C + \sum_{\tau_j \in \Gamma'} C'_j \leq T_0 \), which in turn is equivalent to \( U(\tau_0) + U(\Gamma') \leq 1 \). Thus, the schedulability of \( \Gamma' \) is ensured by Eq. (3).

In any time interval \( L \geq T_0 \) the processing demand of \( \Gamma' \) is given by Eq. (4) whereas the maximum demand due to tasks in \( \Gamma \) equals \( \sum_{\tau_j \in \Gamma'} \left| \frac{T_j}{T_0} \right| C'_j \).

\[
\sum_{\tau_j \in \Gamma'} \left| \frac{T_j}{T_0} \right| C'_j, \tag{4}
\]

As \( \left| \frac{T_j}{T_0} \right| = \left| \frac{T_j T_0}{T_0^2} \right| \), rewriting Eq. (4), we have that

\[
\sum_{\tau_j \in \Gamma'} \frac{L T_j}{T_0} C'_j = \sum_{\tau_j \in \Gamma'} \frac{L T_j}{T_0} C'_j 
\sum_{\tau_j \in \Gamma'} \frac{L T_j}{T_0} C'_j 
\sum_{\tau_j \in \Gamma'} \frac{L T_j}{T_0} C'_j \tag{5}
\]

It follows from Eq. (5) that the processing demand due to tasks in \( \Gamma \) is not greater than that of tasks in \( \Gamma' \). As both task sets are scheduled according to EDF, the schedulability of \( \Gamma' \) implies the schedulability of \( \Gamma \). Therefore, Eq. (3) is a sufficient schedulability condition.

The third schedulability test of interest is given by Theorem 3 and also requires that \( T_0 \) is not greater than the periods of tasks in \( \Gamma \).

**Theorem 3 (Test 3):** Let \( \Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\} \) be a set of tasks scheduled by EDF and let \( \tau_0 \) be the highest priority task in the system such that \( T_0 \leq \min_{\tau_i \in \Gamma}(T_i) \). There is no deadline miss provided that Eq. (6) holds.

\[
\left( \frac{U(\Gamma)}{\min_{\tau_i \in \Gamma}(T_i)} + 1 \right) U(\tau_0) + U(\Gamma) \leq 1 \tag{6}
\]

**Proof:** Let \( t^\varphi \) be the available time to execute the tasks in \( \Gamma \) within a time interval \( L \). From Theorem 8 in [8] it is known that if \( \forall L \geq \min_{\tau_i \in \Gamma}(T_i) \) Eq. (7) holds, then every tasks in \( \Gamma \) meet their deadlines.

\[
L \sum_{\tau_i \in \Gamma} U(\tau_i) \leq t^\varphi \Rightarrow U(\Gamma) \leq \frac{t^\varphi}{L} \tag{7}
\]

The minimum values of \( t^\varphi \) take place when \( \tau_0 \) is periodically activated. Also, the values of \( L \) for minimizing the right-hand side of Eq. (7) occur when the start and ending of the interval \( L \) coincide with the activation and finishing of \( \tau_0 \), respectively. This is because if \( L \) is further increased by \( \epsilon \), \( 0 < \epsilon \leq T_0 - C_0 \), the value of \( t^\varphi \) is also increased by \( \epsilon \). In turn, if the value of \( L \) is decreased by a positive amount \( \epsilon < C_0 \), \( t^\varphi \) is kept constant. In other words, the values of \( L \) to be considered is given by \( L = (k + j)T_0 + C_0 \), where \( k = \left\lfloor \frac{\min_{\tau_i \in \Gamma}(T_i)}{T_0} \right\rfloor \) and \( j \in Z_+ \). In this case, for each time interval of size \( T_0 \), there are \( (T_0 - C_0) \) time units available for executing tasks in \( \Gamma \). which leads to \( t^\varphi = (k + j)(T_0) - C_0 \). Rewriting Eq. (7),

\[
U(\Gamma) \leq \frac{(k + j)(T_0) - C_0}{(k + j)T_0 + C_0} = \frac{(k + j)(T_0 - U(\tau_0)T_0)}{(k + j)T_0 + U(\tau_0)T_0} 
\frac{1 - U(\tau_0)}{1 + \frac{U(\tau_0)}{k + j}} \tag{8}
\]

The right-hand side of Eq. (8) is an increasing function of \( j \). Letting \( j = 0 \) makes Eq. (8) become Eq. (6), as required.}

As can be noted, all the above schedulability tests run in \( O(n) \) and are based on the processor utilization required by the whole task set. We use a different strategy to derive the fourth schedulability test. First, we show that checking the schedulability of a task \( \tau_i \in \Gamma \) taking into consideration that \( \tau_0 \) is executed at the highest priority level can be done via checking the schedulability of a system composed of only two tasks, \( \tau_0 \) and a virtual-task \( \tau'_i \). This latter task is to model the amount of computation resources required by tasks in \( \Gamma \). This result is stated in Theorem 4. Second, we apply response-time analysis to test the schedulability of a virtual two-task systems. If all of them are schedulable, system \( \Gamma \cup \{\tau_0\} \) is also guaranteed to be schedulable.

**Theorem 4:** Let \( \Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\} \) be a set of tasks scheduled by EDF and let \( \tau_0 \) be the highest priority task executed by the system. No task \( \tau_i \in \Gamma \)
misses its deadline provided task \( \tau_i' = (U(\Gamma)T_i, T_i) \) does not miss its deadline when scheduled with \( \tau_0 \) running at the highest priority level.

**Proof:** We show the theorem by demonstrating that a deadline miss of \( \tau_i \) necessarily leads to a deadline miss of \( \tau_i' \).

Assume that \( \tau_i \) misses its deadline at some time \( d \). Also, without loss of generality, assume that no task misses its deadline before \( d \). Let \( t < d \) be the last time at which the processor is idle. If such a time does not exist, let \( t = 0 \). The maximum processing demand within interval \([t, d)\) due to tasks scheduled by EDF is given by

\[
C = \sum_{i=1}^{n} \left\lfloor \frac{d - t}{T_i} \right\rfloor C_i
\]

Now define task \( \tau = (C, (d - t)) \). It is clear that if \( \tau \) and \( \tau_0 \) are scheduled in a fixed-priority manner with \( \tau_0 \) being the highest priority, \( \tau \) also misses its deadline. This is because the system \( \{\tau_0, \tau\} \) demands the same amount of computing resources within interval \([t, d)\) as the demand of \( \{\tau_0\} \cup \Gamma \). We also observe that

\[
U(\tau) = \sum_{i=1}^{n} \left\lfloor \frac{d - t}{T_i} \right\rfloor C_i \leq U(\Gamma)(d - t) = U(\Gamma)
\]

and that \( d \) is a deadline of both \( \tau \) and \( \tau_i' \). As \( U(\tau) \leq U(\Gamma) = U(\tau_i') \), the demand of \( \tau_i' \) during interval \([t, d)\) cannot be smaller than that of \( \tau \), and \( \tau_i' \) only executes when \( \tau_0 \) does not, we conclude that \( \tau_i' \) must also miss its deadline at or before \( d \) when scheduled together with \( \tau_0 \) as the highest priority task.

Theorem 4 implies that testing the schedulability of another system, created from \( \Gamma \), suffices to determine that tasks in the original system may miss their deadlines. This provides interesting ways of checking the schedulability of task sets by checking the schedulability of only two tasks using well known results. For example, Liu and Layland’s schedulability test [1] could be used in the form

\[
U(\tau_0) + U(\Gamma) \leq 2(\sqrt{2} - 1) \tag{9}
\]

Another option is by Bini *et. al* [7],

\[
(U(\tau_0) + 1)(U(\Gamma) + 1) \leq 2 \tag{10}
\]

In both cases, the tests are applied as if the system was composed of only two tasks, one of which consuming \( U(\Gamma) \) of processor. These tests are only sufficient. It is possible to use exact schedulability tests, though. For this purpose, we apply the well known response-time analysis [6]. Although this is a pseudo-polynomial time procedure, it is known that it usually has a very fast convergence time. Being it applied to a system composed of only two tasks, it is indeed a very efficient schedulability test. In summary, the fourth proposed test is given by computing for each task \( \tau_i \in \Gamma \), the worst-case response time for \( \tau_i' = (C_i', T_i') \), where \( C_i' = U(\Gamma)T_i \) and \( T_i' = T_i \). This is an iterative procedure specified as

\[
R_i^k = C_i' + \left\lfloor \frac{R_i^{k-1}}{T_0} \right\rfloor C_0
\]

with \( R_0 = C_i' \). The procedure stops whenever \( R_i^k = R_i^{k-1} \) or \( R_i^k > T_i' \). In this latter case, \( \tau_i' \) is considered non-schedulable at the second priority level when subject to the interference due to \( \tau_0 \).

![Figure 2. Illustration that Test 4 is sufficient but not exact. The original task set, \( \tau_0 = (1, 2), \tau_1 = (0.5, 3), \tau_2 = (0.8, 4) \), is schedulable while task set \( \tau_0 = (1, 2), \tau_1' = (1.1, 3) \) is not, where \( C_i' = T_iU(\{\tau_1, \tau_2\}) \).](image)

It is important to stress that although response time analysis provides an exact schedulability test for the transformed system \( \{\tau_0, \tau_i'\} \), its result serves only as a sufficient condition for task set \( \{\tau_0\} \cup \Gamma \). This is because the non-schedulability of \( \{\tau_0, \tau_i'\} \) does not imply the non-schedulability of \( \{\tau_0\} \cup \Gamma \). To see this consider \( \tau_0 = (1, 2), \tau_1 = (0.5, 3) \) and \( \tau_2 = (0.8, 4) \). As can be noted, task set \( \{\tau_0, \tau_1'\} \) is not schedulable, where \( \tau_1' = (1.1, 3) \). However, the original system \( \{\tau_0, \tau_1, \tau_2\} \) is schedulable, as shown in Fig. 2.

**4. Assessment**

In order to compare the performance of the derived schedulability tests we generated 66,000 task sets with \( n = 2, 4, 8, 16, 32, 64 \) tasks each. The values for the task set utilization were varying between 70% and 100%. For each value of task set utilization 1,000 tasks sets were generated. All these synthetic task sets were generated according to a random task generator described elsewhere [9], a procedure that ensures the uniformity of task set utilization. Task periods were
generated according to a log-uniform integer distribution in the interval \([10, 1000]\), as recommended by other authors [5].

Fig. 3 depicts the behavior of the proposed schedulability tests in terms of *success ratio*, that is, the percentage of task sets accepted as schedulable. For the sake of comparison the performance of the test given by Eq. (10) was also plotted. Eq. (9) was not considered since it leads to a bound of 0.83, which is not better than most values found by the other tests.

As can be seen in the figure, the larger the task set, the better the performance of the schedulability tests. This is because the generated task utilization of each task tends to be lower when \(n\) increases. In particular, the lower the utilization of the highest priority task, the lower its interference in the execution of the other tasks. On average, Eq. (10) behaves worse than Tests 1, 2, 3 and 4. Also, Tests 1 and 3 have similar performance. Note that these tests use a relation between the periods of two tasks only whereas Test 2 inflates the utilization of the task set considering all tasks. As for Test 4, it can be seen that it usually gives better results than the other tests. However, we observe that Tests 1, 2, 3 and 4 do not dominate one another. This motivates combining the proposed tests. Two such combinations are plotted in the graphs, Tests 1, 2 and 3 (plotted as Test 1-3), and Tests 1, 2, 3 and 4 (plotted as Test 1-4). That is, a task set is considered schedulable when it passes through at least one of the tests considered in the combination. Doing so one can get the best out of each schedulability test.

We also compared the proposed tests against an exact one, but with exponential runtime. To do so we chose schedulability test proposed by Burns *et al.* [5]. Although this test was derived in the context of multiprocessor systems, it can be applied for the uniprocessor scheduling model considered in this paper. Fig. 4 summarizes the obtained results. Its \(y\)-axis represents the percentage of feasible task sets that are considered schedulable by the proposed tests. As can
be seen in the figure, the higher the value of \(n\) the more precise is the performance of the proposed tests, which is in line with the results shown in Fig. 3. The exception is for \(n = 2\) and Test 1-4 because Test 4 works as an exact test when the system has only two tasks. It can be also seen that up to values of utilization around 0.95, the performance of the proposed tests is comparable to that of an exact test for \(n \geq 32\) tasks. This is a very good result since the exact test is exponential.

5. Conclusion

We have derived four new sufficient schedulability tests for uniprocessor real-time systems. The considered system is composed of tasks scheduled by EDF which suffer interference of the execution of a high priority task. Experiment results have indicated that the proposed tests have good performance. Schedulable systems that use up to around 95% of processor are identified as such. As the system model considered here is found in practice when urgent routines are not to be delayed by application tasks, the proposed tests have relevance from both theoretical and practical perspective.

Acknowledgment

The authors would like to thank Dr. Konstantinos Bletsas by his valuable comments on an earlier version of this paper. This work was partially funded by FAPESB and CNPq.

References


