Sensitivity Analysis of Task Period for EDF Scheduled Arbitrary Deadline Real-Time Systems

Fengxiang Zhang
Southwest University, China
zhangfx@swu.edu.cn

Alan Burns
University of York, UK
burns@york.ac.uk

Sanjoy Baruah
University of North Carolina at Chapel Hill, USA
baruah@cs.unc.edu

Abstract—The correctness of a real-time system depends on not only the system’s output but also on the time at which results are produced. A hard real-time system is required to complete its operations before all its timing deadlines. For a given task set, it is very useful in an engineering context to know what changes to period (interarrival time) can be made to a task that will deliver a schedulable system. In this paper, we address the sensitivity analysis (parameter calculations) of task period for EDF scheduled systems on a uniprocessor. We prove that a minimum task period can be determined by a single pass of the QPA algorithm. This algorithm provides exact and efficient sensitivity analysis for arbitrary deadline real-time systems. The approaches developed for task parameter computations are therefore as efficient as QPA, and are easily incorporated into a system design support tool.

I. INTRODUCTION

Real-time systems have stringent timing requirements that have to be guaranteed. The primary aim of schedulability analysis is to enable these guarantees to be provided. In the development of a real-time system, schedulability analysis has a much more constructive and informative role. Typically, in the initial design of a real-time system, the application is decomposed into a set of tasks which are assigned the minimum interarrival times (periods) with the worst-case execution time budgets. Unfortunately, it is often the case that periods or execution time of tasks exceed their initial budgets, leading to an unschedulable system. Similarly, with an operational system, it may be necessary to add enhancements which cause the periods of certain tasks to decrease or new tasks to be added. Also performance issues may require task periods to be reduced.

The system developer has to determine if code optimization is needed and to focus effort on those tasks where it will have the most benefit in terms of obtaining a schedulable system. Focused code optimization and reduction can be achieved by performing sensitivity analysis on tasks’ timing characteristics.

Sensitivity analysis is very useful to the design and development of real-time systems. Reducing task period can result in more precise control performance. Changes, reductions, or additions, are usually applied to a specific task. A typical query for an unschedulable system is which task requires the smallest increase in period to deliver a schedulable system.

Most attempts to apply sensitivity analysis (see Section IV for details) have concentrated on fixed priority preemption scheduling and have used a simple branch and bound (binary search) algorithm to iterate down on the borderline values of the parameters that deliver breakdown schedulability. By breakdown schedulability, we mean that any further change to a task’s characteristics (e.g. decreasing period) will result in unschedulability. These search algorithms are expensive to execute as they require the schedulability test to be executed a large number of times as part of the search.

In this paper, we are extending the sensitivity analysis on execution time [17] to task period. We develop sensitivity analysis for task periods for systems scheduled by EDF (earliest deadline first) on a single processor.

According to the EDF scheduling algorithm, at any time, preemption is allowed; an arrived job with an earlier absolute deadline can preempt the execution of a job with a later absolute deadline. When a job completes its execution, the system chooses the pending job with the earliest absolute deadline to execute. The EDF algorithm is most widely studied dynamic priority scheduling policy for real-time systems, and it has been proved by Dertouzos [4] to be optimal among all scheduling algorithms on a uniprocessor. In the sense that if a real-time task set cannot be scheduled by EDF, then this task set cannot be scheduled by any algorithm.

Recent improvements in the efficiency of analysis for EDF (the QPA algorithm is described in Section III) have allowed, efficient sensitivity analysis for task parameter to be undertaken for arbitrary relative deadline systems. In addition, we derive algorithms that directly deliver the borderline parameter values, no form of search or iteration is required. The methods are therefore very efficient and can easily be incorporated into a design support tool that allows the real-time systems engineer to manage change.

II. SYSTEM MODEL

A hard real-time system comprises a set of \( n \) independent real-time tasks \( \tau_1, \tau_2, \ldots, \tau_n \), each task consists of an infinite or finite stream of jobs or requests which must be completed before their deadlines. Let \( \tau_i \) indicate any given task of the system. Each task can be periodic or sporadic.

Periodic tasks. All jobs of a periodic task \( \tau_i \) have a regular interarrival time \( T_i \), we call \( T_i \) the period of the periodic task \( \tau_i \). If a job for a periodic task \( \tau_i \) arrives at time \( t \), then the next job of task \( \tau_i \) must arrive at \( t + T_i \).

978-1-4244-5539-3/10/$26.00 ©2010 IEEE
Sporadic tasks. The jobs of a sporadic task $\tau_i$ arrive irregularly, but they have a minimum interarrival time $T_i$, we call $T_i$ the period of the sporadic task $\tau_i$. If a job of a sporadic task $\tau_i$ arrives at $t$, then the next job of task $\tau_i$ can arrive at any time at or after $t + T_i$.

Each job of task $\tau_i$ requires up to the same worst-case execution time which equals the task $\tau_i$’s worst-case execution (computation) time $C_i$, where $C_i > 0$, and each job of task $\tau_i$ has the same relative deadline which equals the task $\tau_i$’s relative deadline $D_i$; $D_i$ could be less than, equal to, or greater than $T_i$. If a job of task $\tau_i$ arrives at time $t$, the required worst-case execution time $C_i$ must be completed within $D_i$ time units, and the absolute deadline of this job is $d_i = t + D_i$.

Let $\tau_i[C_i,D_i,T_i]$ be the task which needs to be added to an existing task set, or be the task which has its parameters changed as part of sensitivity analysis. We define the task that needs to be added, or be the task which has its parameters changed as part of sensitivity analysis. We define the notation $\tau_i$ to indicate it is the task that has an unknown parameter. We assume that in the absence of $\tau_i$, the other tasks are schedulable, or else it is impossible make the task set schedulable by only changing $\tau_i$’s period.

Let $U_i$ be the utilization of task $\tau_i$ ($U_i = C_i / T_i$), and define $U$ to be the total utilization of the task set, computed by $U = \sum_i U_i$.

The organization of the rest of the paper is as follows. Section III introduces the exact schedulability analysis for EDF systems which will be used in our sensitivity analysis. Section IV describes some literature related to sensitivity analysis. Section V presents our sensitivity analysis approaches to the task parameter calculation. In Section VI, we give the analysis to finding the minimum $T_i$ for task. The conclusions are given in Section VII.

III. EXACT SCHEDULABILITY ANALYSIS FOR EDF

This section describes the previous research results on exact schedulability analysis for EDF scheduling with arbitrary relative deadlines (i.e. $D_i$ unrelated to $T_i$). In 1980, Leung and Merrill [8] noted that a set of periodic tasks is schedulable if and only if all absolute deadlines in the period $[0,\max\{s_i\} + 2H]$ are met, where $s_i$ is the start time of task $\tau_i$, $\min\{s_i\} = 0$, and $H$ is the least common multiple of the task periods. In 1990, Baruah et al [1] extended this condition for sporadic task systems, and showed that the task set is schedulable if and only if: $\forall t > 0$, $h(t) \leq t$, where $h(t)$ is the processor demand function given by:

$$h(t) = \max_{i \geq 1} \left(0, 1 + \frac{i - D_i}{T_i}\right) C_i.$$  (1)

Using the above necessary and sufficient schedulability test, the value of $t$ can be bounded by a certain value.

Theorem 1 [16] An arbitrary deadline task set is schedulable if and only if $U \leq 1$ and

$$\forall t < L^*_u, h(t) \leq t,$$

where

$$L^*_u = \max \{ (D_i - T_i), \ldots, (D_n - T_n) , \sum \frac{(T_i - D_i)U_i}{1 - U} \}.$$  (2)

As the processor demand function can only change at the absolute deadlines of the tasks, only the absolute deadlines require to be checked in an upper bound interval.

In 1996, Spuri [12] and Ripoll et al [11] derived another upper bound for the time interval which guarantees we can find an overflow if the task set is not schedulable. This interval is called the synchronous busy period (the length of the first processor busy period of the synchronous task arrival pattern). However, Ripoll et al [11] only considered the situation where $D_i \leq T_i$.

The length of the synchronous busy period $L_s$ can be computed by the following process [11, 12]:

$$w^p = \sum_{i = 1} C_i,$$  (3)

$$w^{n+1} = \sum_{i = 1} \frac{w^n}{T_i} C_i,$$  (4)

the recurrence stops when $w^{n+1} = w^n$, and then $L_s = w^{n+1}$.

In extensive simulation studies [15, 16] it was nearly always the case that $L^*_u < L_s$.

A. Quick Processor-demand Analysis

In a given interval (i.e. between 0 and $L^*_u$), there can be a very large number of absolute deadlines that need to be checked. This level of computation has been a serious disincentive to the adoption of EDF scheduling in practice. A new much less computation-intensive (but still exact) test known as Quick convergence Processor-demand Analysis (QPA) has been proposed [16].

Let $L$ be upper bound for the schedulability analysis (i.e. $L_u^*$ or $L_s$). Define a failure point to be any time $t$ satisfying $h(t) > t$. QPA works by starting with a value of $t$ close to $L$ and then iterating back through a simple expression toward 0 or the largest failure point in $[0, L]$. It jumps over deadlines that can safely be ignored and hence only a small number of points are checked. For example, a 16 task system that in the previous analysis had to check 858,331 points (deadlines) can, with QPA, be checked at just 12 points. The efficiency of QPA was determined by an extensive set of experiments [15, 16].

Let $d_i$ be an absolute deadline of a job for task $\tau_i$, then $d_i = kT_i + D_i$ for some $k$ such that $k \in N$. The QPA test is given by the following algorithm and theorem.
$t \leftarrow \max \{d_i \mid d_i < L\}$;

while ($t > d_{\text{min}}$)

| if ($h(t) < t$) | $t \leftarrow h(t)$; |
| else if ($h(t) = t$) | $t \leftarrow \max \{d_i \mid d_i < t\}$; |
| else break; // failure deadline is found |

if ($t > d_{\text{min}}$) the task set is unschedulable;
else the task set is schedulable;

Algorithm 1. QPA

Theorem 2 [15] An arbitrary deadline task set is schedulable if and only if $U \leq 1$, and the iterative result of Algorithm 1 is $t \leq d_{\text{min}}$, where $d_{\text{min}} = \min_{i \in \mathcal{S}} \{D_i\}$.

Property 1 [16] QPA is an iterative process to find the largest failure deadline in a given time interval for unschedulable task sets.

Property 2 [16] For an unschedulable task set, the iterative process of Algorithm 1 stops with $t = d_0$, where $d_0$ is the largest deadline in the interval.

Lemma 1 [16] For unschedulable systems, let $d_0$ be a failure deadline found by QPA, then $d_0 < h(d_0) < d^*$, where $d^*$ is the closest deadline larger than $d_0$; i.e.

$$d^* = \min\{d \mid d > d_0\}.$$ 

Observe that from Lemma 1 we can conclude that there is no absolute deadline between $d_0$ and $h(d_0)$.

IV. RELATED WORK

There are a number of papers that have concentrated on sensitivity analysis for fixed priority scheduled systems and have used a simple branch and bound (binary search) algorithm to iteratively down on the optimal values of the parameters which deliver breakdown schedulability. These search algorithms are expensive to execute as they require the schedulability test to be executed a large number of times as part of the search.

The first work to address sensitivity analysis was Lehoczky [7] in 1989. He computed the critical scaling factor as the largest possible decrease/increase in the execution time of all tasks that would deliver a schedulable system that was on the borderline of unschedulability. His work was for fixed priority preemptive scheduling with rate monotonic priority assignment.

Improvements to Lehoczky’s analysis were made by Katcher and Yerraballi [6, 14], considering parameters other than execution time, producing better bounds [13], and using exact response-time analysis [9]. These methods all used binary search to identify the borderline parameter values. Further improvements were introduced by Regehr [10] and Bini et al [3]. Tool sets aimed at fixed priority scheduling also incorporated sensitivity analysis – for example the MAST tool suite [5]. Most of these approaches were concerned fixed priority scheduling with uniprocessor systems.

In the following sections of this paper, we will present exact and efficient sensitivity analysis of task period for EDF scheduled systems without any restrictions (i.e. each task’s period can be less than, equal to, or greater than its relative deadline).

V. APPROACH TO SENSITIVITY ANALYSIS

QPA tests the schedulability of a task set quickly by a single iterative process, based on this property, we can exploit it to deliver the optimal task parameters. The essential idea of our task parameter computations is to start with an unschedulable task set and use QPA to test it, when a failure point is found, we change the task parameter at this point and let the iterative process continue. We prove that whenever a task parameter is changed at time $t$; values greater than $t$ need not to be reassessed; the algorithm can continue from $t$ towards $0$.

At the beginning of a parameter calculation, we let the initial $T_s$ be the minimum possible value. As the system is unschedulable with the initial task parameters, then a failure deadline $d_0$ must be found; we change the parameter at $d_0$, and then let QPA continue iterating back from $d_0$ until $t \leq D_s$. Due to the high efficiency of QPA in finding a failure time point for an unschedulable system, and the quick convergence property of QPA for a schedulable system, we can get the optimal task parameters quickly by this approach.

Lemma 2 After the value of $T_s$ is increased, there is no failure point in $[h(d_0), L]$.

Proof. From Property 1, $\forall t \in [h(d_0), L], h(t) \leq t$. Let $h(t)$ be the function after the parameter change, we have $h(t) \leq h(t)$, so $\forall t \in [h(d_0), L], h(t) \leq h(t) \leq t$. □

Figure 1. Approach to the sensitivity analysis

Let $d_0$ be a missed deadline found by QPA. From Property 1, $d_0$ is the largest failure deadline, hence there is no failed point in the interval $[h(d_0), L]$. From Lemma 2, after the increase of $T_s$, we still have no failed point in the interval $[h(d_0), L]$, as shown as Figure 1.

Property 3 Any $t$ in the interval $[d_0, h(d_0)]$ is also a failure point before the parameter change.

Proof. From Lemma 1, there is no absolute deadline in the interval $[d_0, h(d_0)]$. For any $t$ in the interval $[d_0, h(d_0)]$, we have $h(t) = h(d_0) > t$. □

From Lemma 2 and Property 3, when a failure deadline $d_0$ is found, we only need to concentrate on the interval $[d_0, h(d_0)]$. If there is no failure point in $[d_0, h(d_0)]$ after the parameter change, then there is also no failure point in the interval $[d_0, L]$.  

25
Let $L'$ be the new upper bound after the parameter change, as $L'$ is always less than or equal to the initial value of $L$, the interval $[d_0, L']$ is also safe and need not be rechecked.

As any $t$ in the interval $[d_0, h(d_0))$ is a failure point, we need to increase $T$, to make sure there is no failure in this time interval. Hence in $[d_0, h(d_0))$, we have to find the maximum required $T$, so that for any $t \geq d_0$, $h(t) \leq t$. In the following section, we will address this issue.

**VI. MINIMUM PERIOD CALCULATION**

This section discuss the situation when the value of $C_5$ and $D_5$ are fixed, we need to compute the minimum value of $T$ to keep the task set schedulable. At the beginning of the calculation, we set $T = C_5$ and decreased to $U = 1$, that is

$$T_i = C_i / (1 - \sum_{i=1}^{n} C_i / T_i)$$

When a missed deadline $d_0$ is found with this initial $T$’s value, following the discussions of Section V, we need to calculate the maximum required $T_i$, to make sure there is no failure point in this interval. Then $t$ continues iterating back from $D_5$. When $t < D_5$, $\tau_i$ has no contribution to $h(t)$, as $\tau_1, \tau_2, ..., \tau_n$ is schedulable, the iterative process ends with $t < D_5$.

$$h(t) = \max_{i=1}^{n} \left[ \frac{t - D_i}{T_i} \cdot C_i + \frac{D_i}{T_i} \cdot C_i \right]$$  \hspace{1cm} (5)

**Theorem 3** Let $d_0$ be a failure deadline found by QPA, and

$$M = \sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]$$

define $t_0 = \min \left\{ t | t = kC_i + M \land t > d_0 \right\}$, where $k \in N$.

When $M > D_i$, then there will be no failure deadline in the interval $[d_i, h(d_i))$ if and only if

$$T_i > (h(d_i) - D_i) / \left[ C_i \cdot (h(d_i) - D_i) / C_i \right]$$  \hspace{1cm} (6)

When $M \leq D_i$, there will be no failure deadline in $[d_i, h(d_i))$ if and only if

$$T_i > (h(d_i) - D_i) / \left[ C_i \cdot (h(d_i) - D_i) / C_i \right]$$  \hspace{1cm} (7)

**Proof.** Let $t_0$ be any failure point, if we need $h(t_0) \leq t_0$, from equation (5):

$$\sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i + \frac{D_i}{T_i} \cdot C_i \right] \leq t_0$$  \hspace{1cm} (5)

$$\Leftrightarrow \left[ \frac{t_0 - D_i}{T_i} \right] \leq \sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]$$

$$\Leftrightarrow \left[ \frac{t_0 - D_i}{T_i} \right] \leq \left[ \frac{\max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]}{C_i} \right] - 1$$

$$\Rightarrow t_0 - D_i < \left[ \frac{\max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]}{C_i} \right] - 1 + 1$$

$$\Rightarrow t_0 - D_i = \left[ \frac{\max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]}{C_i} \right]$$

From Lemma 1, when $t \in [d_i, h(d_i))$:

$$\sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right] = \sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]$$

let $M = \sum_{i=1}^{n} \max \left[ \frac{d_i - T_i}{T_i} \cdot C_i \right]$

then inequality (8) is changed to:

$$T_i > (t_0 - D_i) / \left[ C_i \cdot (t_0 - D_i) / C_i \right]$$

Let

$$f(t) = (t_0 - D_i) / \left[ t_0 - M \cdot C_i \right]$$

When $kC_5 + M \leq t < (k+1)C_5 + M$, since

$$\left[ t_0 - M \cdot C_i \right]$$

is only changed when $t = kC_5 + M - \varepsilon$, where $k \in N$, and let $\varepsilon > 0$ be an infinitesimally small number, $f(t)$ is increasing with $t$, and $f(t)$ gets the maximum when $t = (k+1)C_5 + M - \varepsilon$.

When $t = kC_5 + M$:

$$f(t) = (t_0 - D_i) / \left[ t_0 - M \cdot C_i \right]$$

then:

$$\frac{df(t)}{dt} = \frac{d(M)}{dt} \cdot C_i / (t_0 - D_i) + (t_0 - D_i) / \left[ t_0 - M \cdot C_i \right]$$

when $M > D_i$, $f(t)$ is a decreasing function of $t$, and when $M \leq D_i$, $f(t)$ is an increasing function of $t$.

From the above discussions, when $M \leq D_i$, $f(t)$ is an increasing function of $t$ in the whole interval $[d_i, h(d_i))$, then the maximum $f(t)$ is obtained when $t = h(d_i) - \varepsilon$. When $M > D_i$, and $t = kC_5 + M$, $f(t)$ is a decreasing function of $t$; when $kC_5 + M \leq t < (k+1)C_5 + M$, the maximum $f(t)$ is obtained when $t = (k+1)C_5 + M - \varepsilon$. Therefore if $M > D_i$, $f(t)$ gets the maximum when $t = t_0 - \varepsilon$; and when $M \leq D_i$, the maximum $f(t)$ is obtained when $t = h(d_i) - \varepsilon$. □
In Theorem 3, the formulas for $T_s$ could lead to a divide by 0. In the following, we demonstrate that if the denominator equals to 0, then no value of $T_s$ will lead to a schedulable system.

Lemma 3 When $M > D_s$, if \[ \frac{t_s - \varepsilon - M}{C_s} = 0, \]
then it is impossible to add such a task $t_s$ to let the new task set be schedulable.

Proof. \[
\frac{t_s - \varepsilon - M}{C_s} = \left[ \frac{d_b - M}{C_s} \right] \frac{C_s + M - \varepsilon - M}{C_s}
\]
\[
= \left[ \frac{d_b - M}{C_s} \right] - \frac{\varepsilon}{C_s} = \left[ \frac{d_b - M}{C_s} \right] - 1.
\]
Then \[
\frac{t_s - \varepsilon - M}{C_s} = 0 \Leftrightarrow \left[ \frac{d_b - M}{C_s} \right] - 1 = 0
\]
\[
\Leftrightarrow \left[ \frac{d_b - M}{C_s} \right] = 1 \Leftrightarrow d_b - M \leq C_s.
\]
If \[ \frac{d_b - M}{C_s} = 1 \] then \[ M = d_b - C_s, \]
and \[ \frac{t_s}{t_b} = \frac{C_s + M}{C_s + d_b - C_s} = d_b, \]
this contradicts the condition \( t_s > d_b \), therefore \( d_b - M < C_s \). (9)

Since tasks $t_1, t_2, \ldots, t_{s+1}$ are schedulable, $d_b \geq M$, and we have \[ d_b \geq M > D_s, \]
then \( h(d_b) \geq M + C_s \), from inequality (9), \( h(d_b) \geq M + C_s > d_b \). \( \Box \)

Lemma 4 If \[
\frac{h(d_b) - \varepsilon - M}{C_s} = 0,
\]
then it is is impossible to add such a task $t_s$ to let the task set be schedulable.

Proof. \[
\frac{h(d_b) - \varepsilon - M}{C_s} = 0 \Leftrightarrow h(d_b) - \varepsilon - M < C_s
\]
\[
\Leftrightarrow M + C_s > h(d_b) - \varepsilon > d_b. \quad (10)
\]
Since $M \leq d_b$, and $d_b$ is a failure point when $t_s$ is included, then \( d_b \geq D_s \), and at least one $C_s$ must be counted in $h(d_b)$; we have \( h(d_b) \geq M + C_s \), from inequality (10), \( h(d_b) \geq M + C_s > d_b. \) \( \Box \)

\[
T_s = C_s / \left( 1 - \sum_{i=1}^{s+1} C_i \right); \quad (11)
\]
\[
t \gets \max \{ d_i | d_i < L_s \}; \quad (12)
\]

while \( (t \geq D_s) \)
\[
\{ \text{if} (h(t) < t) \quad t \gets h(t); \}
\]
\[
\text{else if} \ (h(t) = t) \quad t \gets \max \{ d_i | d_i < t \}; \}
\]
\[
\text{else} \quad / \text{recalculate parameter $T_s$}
\]
\[
\{ t_s \leftarrow \frac{t + M}{C_s} \}; \quad \text{//calculate $t_s$}
\]
\[
\text{if} (t_s = t) \quad t_s \leftarrow t_s + C_s;
\]
if ($M > D_s$)
\[
\{ \text{if} \left( \frac{t_s - \varepsilon - M}{C_s} \right) \neq 0 \}
\]
\[
T_s \leftarrow \frac{t_s - \varepsilon - M}{C_s} + \varepsilon; \quad (13)
\]
else Return it is impossible to add such task;
\[
\text{else if} \left( \frac{h(t) - \varepsilon - M}{C_s} \right) \neq 0
\]
\[
T_s \leftarrow h(t) - \varepsilon - M + \varepsilon; \quad (13)
\]
else Return it is impossible to add such task;
\[
\}
\]

Algorithm 2. Minimum $T_s$ calculations

Since $L_s$ is not monotonic decreasing when $T_s$ is increased, we use $L_s$ as the upper bound for the initial value of $t$.

According to Theorem 3, Lemma 3, and Lemma 4, the calculation of $T_s$ is given by Algorithm 2.

VII. CONCLUSION

In the design and implementation of a real-time system, it is useful to identify the conditions necessary for a given set of tasks to meet all their deadlines when executed on a specified hardware platform. It is often the case that we need to know what is the minimum task period that will result in a system that is borderline schedulable.

In this paper, we answered the above question by developing a simple iterative process. We prove that the minimum task period can be computed by a single pass of the QPA algorithm. There is no restriction on the task parameters; there is no additional search or iterations required by the calculations, hence the approaches are as efficient as QPA, and they can easily be incorporated into a design support tool.

As a future work, we plan to extend this sensitivity analysis to task relative deadline parameters.

VIII. ACKNOWLEDGEMENTS

This work was funded by the Southwest University Science Foundation (project No. SWU110006).

REFERENCES


