2010 Pacific Rim International Symposium on Dependable Computing

Sensitivity Analysis of the Minimum Task Period for Arbitrary Deadline Real-Time Systems

Fengxiang Zhang
Department of Computer Science
Southwest University
Chongqing, China
zhangfx@ieee.org

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

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

Abstract—**The most important character of real-time systems is that they have stringent timing deadlines that must be guaranteed. A hard real-time system is required to complete its operations before all its timing deadlines. For a given task set, it is useful in an engineering context to know what changes to period can be made to a task that will deliver a schedulable system. In this paper, we develop the sensitivity analysis 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; an improved scheme is presented by using different initial values of the period. The approaches developed for sensitivity analysis of task period are therefore as efficient as QPA, and are easily incorporated into a system design support tool.**

Keywords: real-time and embedded systems; system design and control; performance and reliability

1. INTRODUCTION

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. 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 times 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 4 for details) have concentrated on fixed priority preemptive 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, based on the initial work of [16] (ICCSIT), we develop sensitivity analysis of the optimal (minimum) task period for systems scheduled by EDF (earliest deadline first) on a single processor. This paper has the following significant improvements to the approach that was provided in [16]:

1) the infinitesimally small value ε is removed from the task period computations, hence, the calculated result is optimal.
2) this paper addresses the problem when the total utilization of the task set is 1, therefore, more efficient start values for the task period computations are presented.

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 [3] to be optimal among all scheduling algorithm 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 al-

978-0-7695-4289-8/10 $26.00 © 2010 IEEE
DOI 10.1109/PRDC.2010.16

gorithm.

The QPA algorithm is described in Section 3 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.

2. SYSTEM MODEL

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

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

Sporadic tasks. The jobs of a sporadic task τ_i arrive irregularly, but they have a minimum interarrival time T_i, we call T_i the period of the sporadic task τ_i. If a job of a sporadic task τ_i arrives at t, then the next job of task τ_i can arrive at any time at or after $t+T_1$.

Each job of task τ_i requires up to the same worst-case execution time which equals the task τ_i's worst-case execution (computation) time C_i, where $C_i > 0$, and each job of task τ_i has the same relative deadline which equals the τ_i's relative deadline D_i; D_i could be less than, equal to, or greater than T_i. If a job of task τ_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 $\tau_x\{C_x, D_x, T_x\}$ 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 other tasks in the system to be $\tau_1, \tau_2, ..., \tau_{n-1}$. Hence the additional task, τ_x could also be denoted as τ_n, however, we retain the notation τ_x to indicate it is the task that has an unknown parameter. We assume that in the absence of τ_x, the other tasks are schedulable, or else it is impossible to make the task set schedulable by only changing τ_x's period.

Let U_i be the utilization of task τ_i ($U_i = C_i / T_i$), and define U to be the total utilization of the task set, computed by $U = \sum_{i=1}^{n} U_i$.

The organization of the rest of the paper is as follows. Section 3 introduces the exact schedulability analysis for EDF systems which will be used in our sensitivity analysis. Section 4 describes some literature related to sensitivity analysis. Section 5 presents our sensitivity analysis approaches to the task parameter calculation. In Section 6, we give the analysis to finding the minimum T_x for task. The conclusions are given in Section 7.

3. EXACT SCHEDULABILITY ANALYSIS

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 [7] 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 τ_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: $\forall t > 0$, $h(t) \le t$, where $h(t)$ is the processor demand function given by:

$$h(t) = \sum_{i=1}^{n} \max\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\} C_i \ . \qquad (1)$$

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

Theorem 1 [15] An arbitrary deadline task set is schedulable if and only if $U \le 1$ and

$$\forall t < L_a^*,\ h(t) \le t\ ,$$

where

$$L_a^* = \max\{(D_1 - T_1), ..., (D_n - T_n), \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1-U}\}\ . \qquad (2)$$

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 [11] and Ripoll et al [10] 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. [10] only considered the situation where $D_i \le T_i$.

The length of the synchronous busy period L_b can be computed by the following process [10, 11]:

$$w^0 = \sum_{i=1}^{n} C_i\ , \qquad (3)$$

$$w^{m+1} = \sum_{i=1}^{n} \left\lceil \frac{w^m}{T_i} \right\rceil C_i\ , \qquad (4)$$

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

A. *Quick Processor-demand Analysis*

In a given interval (e.g. between 0 and L_a^*), 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 [15].

Let L be upper bound for the schedulability analysis(i.e. L_a^* or L_b). 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 [14, 15].

Let d_i be an absolute deadline of a job for task τ_i, then $d_i = kT_i + D_i$ for some k, $k \in N$. The QPA test is given by the following algorithm and theorem.

```
$t \leftarrow \max\{d_i \mid d_i < L\}$;
**while** ($t > d_{\min}$)
  {**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** ($t > d_{\min}$)   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 $U \le 1$, and the iterative result of Algorithm 1 is $t \le d_{\min}$, where $d_{\min} = \min_{1\le i\le n}\{D_i\}$.

Property 1 [15] QPA is an iterative process to find the largest failure deadline in a given time interval for unschedulable task sets.

4. RELATED WORK

There are a number of papers that have concentrated on sensitivity analysis for fixed priority scheduled systems, and they have used a simple binary search algorithm to iterate 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 [6] 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 [5, 13]: considering parameters other than execution time, producing better bounds [12], and using exact response-time analysis [8]. These methods all used binary search to identify the borderline parameter values. Further improvements were introduced by Regehr [9] and Bini et al. [2]. Tool sets aimed at fixed priority scheduling also incorporated sensitivity analysis – for example the MAST tool suite [4]. Most of these approaches were concerned fixed priority scheduling with uniprocessor systems.

Zhang et al [16] presented an efficient sensitivity analysis of task period for EDF scheduled systems, however, this analysis is sub-optimal and the calculated result cannot get to the minimum. In Section 6, we will present the minimum task period calculations for EDF systems with arbitrary relative deadlines.

5. APPROACH TO SENSITIVITY ANALYSIS

This section describes our approach to task period computations [16]. The essential idea of our sensitivity analysis is to start with an unschedulable task set and use QPA to test it, when a failure point is found, we change the task period at this point and let the iterative process continue. We prove that whenever a task period is increased at time t, values greater than t need not to be reassessed; the algorithm can continue from t towards 0.

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 $[h(d_0), L)$.

Lemma 1 [16] After the value of T_x is increased, there is no failure point in $[h(d_0), L]$.

Property 2 [17] Any t in the interval $[d_0, h(d_0))$ is also a failure point before the parameter change.

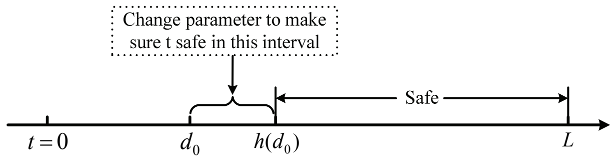

Figure 1. Task parameter change at d_0

From Lemma 1, after the increase of T_x, we still have no failed point in the interval $[h(d_0), L)$, as shown in Figure 1.

From Lemma 1 and Property 2, 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 $[d_0, L]$.

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_x, 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_x so that for any $t \geq d_0$, $h(t) \leq t$. In the following section, we will address this issue.

6. MINIMUM PERIOD CALCULATION

This section discuss the situation when the value of C_x and D_x are given, we need to compute the minimum value of T_x to keep the task set schedulable. This means task τ_x's period can be optimally decreased to a certain value T_x while the task set is still schedulable, and if T_x is further decreased by any infinitesimally small value, then the task set becomes unschedulable.

When a missed deadline d_0 is found with an initial T_x's value, following the discussions of Section 5, we need to calculate the maximum required T_x in $[d_0, h(d_0))$ to make sure there is no failure point in this interval. Then t continues iterating back from d_0.

Theorem 3 [16] Let ε be an infinitesimally small value, d_0 be a failure deadline found by QPA, and

$$M = \sum_{i=1}^{n-1} \max\{0, \left\lfloor \frac{d_0 + T_i - D_i}{T_i} \right\rfloor\} C_i ;$$

define $t_\kappa = \min\{t \mid t = kC_x + M \wedge t > d_0\}$, where $k \in N$.

When $M > D_x$, then there will be no failure deadline in the interval $[d_0, h(d_0))$ if and only if

$$T_x > (t_\kappa - \varepsilon - D_x) / \left\lfloor \frac{t_\kappa - \varepsilon - M}{C_x} \right\rfloor. \qquad (5)$$

When $M \leq D_x$, there will be no failure deadline in $[d_0, h(d_0))$ if and only if

$$T_x > (h(d_0) - \varepsilon - D_x) / \left\lfloor \frac{h(d_0) - \varepsilon - M}{C_x} \right\rfloor. \qquad (6)$$

Theorem 3 provides an approach to recalculate the task period when a failure deadline is found by the QPA algorithm. However, the calculation is sub-optimal, this means the computed task period is not the minimum, and how close is the value of T_x to the minimum depends on the value selection of ε. In the remainder this section, we show that the infinitesimally small value ε can be removed from the algorithm for the T_x calculations.

Lemma 2 If $\left(\frac{t_\kappa - M}{C_x} - 1 \right) = 0$,

then it is impossible to add such a task τ_x to let the task set be schedulable.

Proof. $\left(\frac{t_\kappa - M}{C_x} - 1 \right) = 0$

$$\Leftrightarrow \quad t_\kappa - M = C_x$$
$$\Leftrightarrow \quad C_x + M = t_\kappa > d_0 \qquad (7)$$

Since tasks $\tau_1, \tau_2, ..., \tau_{n-1}$ are schedulable, and d_0 is a failure point when task τ_x is accounted, there must be at least one C_x included in $h(d_0)$, and we have $h(d_0) \geq M + C_x$, from inequality (7), $h(d_0) \geq M + C_x > d_0$. □

Lemma 3 When it is possible to add task τ_x to let the task set be schedulable, then

$$\lim_{\varepsilon \to 0^+} \left((t_\kappa - \varepsilon - D_x) / \left\lfloor \frac{t_\kappa - \varepsilon - M}{C_x} \right\rfloor + \varepsilon \right)$$

$$= (t_\kappa - D_x) / \left(\frac{t_\kappa - M}{C_x} - 1 \right).$$

Proof. Since $t_\kappa = \left\lceil \frac{d_0 - M}{C_x} \right\rceil C_x + M \quad \Rightarrow$

$$\frac{t_\kappa - M}{C_x} = \left\lceil \frac{d_0 - M}{C_x} \right\rceil = k \in N \text{, therefore}$$

$$(t_\kappa - \varepsilon - D_x) / \left\lfloor \frac{t_\kappa - \varepsilon - M}{C_x} \right\rfloor + \varepsilon$$

$$= (t_\kappa - \varepsilon - D_x) / \left\lfloor \frac{t_\kappa - M}{C_x} - \frac{\varepsilon}{C_x} \right\rfloor + \varepsilon$$

$$= (t_\kappa - \varepsilon - D_x) / \left(\frac{t_\kappa - M}{C_x} - 1 \right) + \varepsilon .$$

Then $\lim_{\varepsilon \to 0^+} \left((t_\kappa - \varepsilon - D_x) / \left(\frac{t_\kappa - M}{C_x} - 1 \right) + \varepsilon \right)$

$$= \lim_{\varepsilon\to 0^+}(t_\kappa - D_x)/\left(\frac{t_\kappa - M}{C_x}-1\right)-\varepsilon/\left(\frac{t_\kappa - M}{C_x}-1\right)+\varepsilon .$$

From Lemma 3, $\frac{t_\kappa - M}{C_x} \neq 1$, then we have $t_\kappa - M \geq 2C_x$, and $\frac{t_\kappa - M}{C_x} \geq 2$, therefore

$$\lim_{\varepsilon\to 0^+}(t_\kappa - D_x)/\left(\frac{t_\kappa - M}{C_x}-1\right)-\varepsilon/\left(\frac{t_\kappa - M}{C_x}-1\right)+\varepsilon$$

$$= (t_\kappa - D_x)/\left(\frac{t_\kappa - M}{C_x}-1\right). \qquad \square$$

Lemma 4 If $\frac{h(d_0)-M}{C_x}-1=0$,

then it is impossible to add such a task τ_x to let the task set be schedulable.

Proof. $\frac{h(d_0)-M}{C_x}=1 \Leftrightarrow h(d_0)-M = C_x$

$\Leftrightarrow \quad M + C_x = h(d_0) > d_0 .$ (8)

Since $M \leq d_0$, and d_0 is a failure point when τ_x is included, then $d_0 \geq D_x$, and at least one C_x must be counted in $h(d_0)$, we have $h(d_0) \geq M + C_x$, from inequality (8), $h(d_0) \geq M + C_x > d_0$. $\square$

Lemma 5 When it is possible to add task τ_x to let the task set be schedulable, then

$$\lim_{\varepsilon\to 0^+}\left((h(d_0)-\varepsilon-D_x)/\left\lfloor\frac{h(d_0)-\varepsilon-M}{C_x}\right\rfloor+\varepsilon\right)$$

$$= (h(d_0)-D_x)/\left(\frac{h(d_0)-M}{C_x}-1\right).$$

Proof. Since $h(d_0) = M + kC_x$, then

$$(h(d_0)-\varepsilon-D_x)/\left\lfloor\frac{h(d_0)-\varepsilon-M}{C_x}\right\rfloor+\varepsilon$$

$$= (h(d_0)-\varepsilon-D_x)/\left\lfloor\frac{h(d_0)-M}{C_x}-\frac{\varepsilon}{C_x}\right\rfloor+\varepsilon$$

$$= (h(d_0)-\varepsilon-D_x)/\left(\frac{h(d_0)-M}{C_x}-1\right)+\varepsilon .$$

Since $h(d_0) = kC_x + M$, we have $\frac{h(d_0)-M}{C_x} \geq 1$. From Lemma , $\frac{h(d_0)-M}{C_x} \neq 1$, then $h(d_0)-M \geq 2C_x$, and $\frac{h(d_0)-M}{C_x} \geq 2$, therefore

$$\lim_{\varepsilon\to 0^+}\left((h(d_0)-\varepsilon-D_x)/\left\lfloor\frac{h(d_0)-\varepsilon-M}{C_x}\right\rfloor+\varepsilon\right)$$

$$= \lim_{\varepsilon\to 0^+}\left((h(d_0)-D_x)/\left(\frac{h(d_0)-M}{C_x}-1\right)-\varepsilon/\left(\frac{h(d_0)-M}{C_x}-1\right)+\varepsilon\right)$$

$$= (h(d_0)-D_x)/\left(\frac{h(d_0)-M}{C_x}-1\right). \qquad \square$$

From Lemma 3 and Lemma 5, then Theorem 3 for the T_x computation can be changed to Theorem 4.

Theorem 4 Let d_0 be a failure deadline found by QPA, and

$$M = \sum_{i=1}^{n-1}\max\{0,\left\lfloor\frac{d_0+T_i-D_i}{T_i}\right\rfloor\}C_i ;$$

define $t_\kappa = \min\{t \mid t = kC_x + M \wedge t > d_0\}$, where $k \in N$.

When $M > D_x$, there will be no failure deadline in the interval $[d_0, h(d_0))$ if and only if

$$T_x \geq (t_\kappa - D_x)/\left(\frac{t_\kappa - M}{C_x}-1\right).$$

When $M \leq D_x$, there will be no failure deadline in $[d_0, h(d_0))$ if and only if

$$T_x \geq (h(d_0)-D_x)/\left(\frac{h(d_0)-M}{C_x}-1\right).$$

Proof. From Lemma 3 and equation (5):

$$T_x > (t_\kappa - \varepsilon - D_x)/\left\lfloor\frac{t_\kappa-\varepsilon-M}{C_x}\right\rfloor$$

$$\Leftrightarrow \quad T_x \geq (t_\kappa - \varepsilon - D_x) / \left\lfloor \frac{t_\kappa - \varepsilon - M}{C_x} \right\rfloor + \varepsilon$$

$$\Leftrightarrow \quad T_x \geq (t_\kappa - D_x) / \left(\frac{t_\kappa - M}{C_x} - 1 \right).$$

From Lemma 5 and equation (6):

$$T_x > (h(d_0) - \varepsilon - D_x) / \left\lfloor \frac{h(d_0) - \varepsilon - M}{C_x} \right\rfloor$$

$$\Leftrightarrow \quad T_x \geq (h(d_0) - \varepsilon - D_x) / \left\lfloor \frac{h(d_0) - \varepsilon - M}{C_x} \right\rfloor + \varepsilon$$

$$\Leftrightarrow \quad T_x \geq (h(d_0) - D_x) / \left(\frac{h(d_0) - M}{C_x} - 1 \right).$$

Then the conclusion can be obtained from Theorem 3. □

At the beginning of the sensitivity analysis, we can decrease the task period to the minimum so that $U = 1$. However, when the total utilization of the task set is equal to 1, the upper bound L_a^* is undefined (divide by 0), and the L_b is the least common multiple of all task periods, thus it could become very large. A more efficient approach is to let T_x be decreased so that $U = 1 - \Delta$ at the start of the computations, where Δ is a very small given value (e.g. $\Delta = 0.02$), hence, the total utilization is very close to 1, then the initial value

$$T_x = C_x / (1 - \Delta - \sum_{i=1}^{n-1} \frac{C_i}{T_i}).$$

If a failure is found with this initial value, then the task period parameter change is optimal (i.e. the minimum T_x is calculated).

If no missed deadline is found with this value, this means T_x is still not the minimum value, then we can decrease the value of T_x, and start the iteration again with an initial value so that the total utilization is more closer to 1 (e.g. let $\Delta = 0.01$). If there is still no failure deadline, then further decrease the initial value with $U = 1$.

```
$\Delta \leftarrow 0.98$;  $find \leftarrow 0$;
**again:**
$T_x = C_x / (1 - \Delta - \sum_{i=1}^{n-1} \frac{C_i}{T_i})$;          (9)
$t \leftarrow \max\{d_i \mid d_i < L_b\}$;
*while* ( $t \geq d_{\min}$ )
  { *if* ( $h(t) < t$ )  $t \leftarrow h(t)$;
    *else if* ( $h(t) = t$ )  $t \leftarrow \max\{d_i \mid d_i < t\}$;
    *else*                    //recalculate the period
      { *if* ( $M > D_x$ )
          { $t_\kappa \leftarrow \left( \left\lfloor \frac{t - M}{C_x} \right\rfloor + 1 \right) C_x + M$;
            *if* ( $\frac{t_\kappa - M}{C_x} - 1 \neq 0$ )
              { $T_x \leftarrow (t_\kappa - D_x) / \left( \frac{t_\kappa - M}{C_x} - 1 \right)$;          (10)
                $find \leftarrow 1$; }
            *else*  return it is impossible to add such task;
          }
        *else*
          {*if* ( $\frac{h(t) - M}{C_x} - 1 \neq 0$ )
              { $T_x \leftarrow (h(t) - D_x) / \left( \frac{h(t) - M}{C_x} - 1 \right)$;          (11)
                $find \leftarrow 1$; }
           *else*  return it is impossible to add such task;
          }
      }
  }
*if* ( $\Delta \neq 1 \wedge find = 0$ )
//iterate again with a smaller $T_x$ initial value
  { $\Delta \leftarrow \Delta + 0.01$;
    *goto* again;
  }
```

Algorithm 2. Minimum T_x calculation

In the algorithm, we can set a variable '*find*' to indicate if there is a missed deadline to be found. Since L_a^* is not monotonic decreasing when T_x is increased, we use L_b as the upper bound for the initial value of t.

According to Theorem 4, Lemma 2 and Lemma 4, the minimum T_x calculation is given by Algorithm 2.

A. *Illustrative Example*

This section gives two examples to illustrate the work of Algorithm 2. The first example comprises 4 tasks, and the existing task set comprises 3 tasks, their parameters an task τ_x's execution time and relative deadline are as follows.

Example 1.

Task	Execution Time	Relative Deadline	Period
τ_1	2	12	11
τ_2	34	86	89
τ_3	65	196	312
τ_x	26	128	?

The minimum T_x is calculated by the following.

1. Calculate the initial value of T_x by equation (9) (Δ =0.02), $T_x = 125.104652$.
2. Calculate the upper bound by equations (3)(4), $L_b = 610$. Hence $\max\{d_i \mid d_i < L_b\} = 606$.
3. Then start the QPA iteration:
   1) $t = 606$, $h(t) = 548$;
   2) $t = 548$, $h(t) = 536$;
   3) $t = 536$, $h(t) = 534$;
   4) $t = 534$, $h(t) = 534$;
   5) $t = 531$, $h(t) = 534$;
   As $M = 430 > D_x$, recalculate T_x by equation (10), $T_x = 135.333328$.
   6) $t = 531$, $h(t) = 508$;
   7) $t = 508$, $h(t) = 470$;
   8) $t = 470$, $h(t) = 397$;
   9) $t = 397$, $h(t) = 325$;
   10) $t = 325$, $h(t) = 277$;
   11) $t = 277$, $h(t) = 269$;
   12) $t = 26$, $h(t) = 267$;
   13) $t = 267$, $h(t) = 267$;
   14) $t = 265$, $h(t) = 267$;
   As $M = 215 > D_x$, recalculate T_x by equation (10), $T_x = 139.000000$.
   15) $t = 265$, $h(t) = 241$;
   16) $t = 241$, $h(t) = 201$;
   17) $t = 201$, $h(t) = 195$;
   18) $t = 195$, $h(t) = 128$;
   19) $t = 128$, $h(t) = 82$;
   20) $t = 82$, $h(t) = 14$;
   21) $t = 14$, $h(t) = 2$;
   As $t = 2 < d_{\min}$, where $d_{\min} = 12$, the iteration stops.

Since a failure deadline is found during the iteration, the minimum value is given by $T_x = 139$.

Example 2.

Task	Execution Time	Relative Deadline	Period
τ_1	4	11	16
τ_2	5	16	20
τ_3	8	26	40
τ_x	3	14	?

The minimum T_x is calculated by the following.

1. Calculate the initial value of T_x by equation (9) (Δ =0.02), $T_x = 10.714285$.
2. Calculate the upper bound by equations (3)(4), $L_b = 80$. $\max\{d_i \mid d_i < L_b\} = 78.285706$.
3. Then start the QPA iteration:
   1) $t = 78.285706$, $h(t) = 74$;
   2) $t = 74$, $h(t) = 65$;
   3) $t = 65$, $h(t) = 54$;
   4) $t = 54$, $h(t) = 42$;
   5) $t = 42$, $h(t) = 35$;
   6) $t = 35$, $h(t) = 27$;
   7) $t = 27$, $h(t) = 27$;
   8) $t = 26$, $h(t) = 23$;
   9) $t = 23$, $h(t) = 12$;
   10) $t = 12$, $h(t) = 4$;

Since there is no failure deadline found during the iteration, start the iteration again with $\Delta = 0.01$:

1. Calculate the initial value of T_x by equation (9) (Δ =0.01), $T_x = 10.344827$.
2. Calculate the upper bound by equations (3)(4), $L_b = 80$. $\max\{d_i \mid d_i < L_b\} = 76.068962$.
3. Then start the QPA iteration:
   1) $t = 76.068962$, $h(t) = 77$;
   As $M = 56 > D_x$, recalculate T_x by equation (10), $T_x = 10.5$.
   2) $t = 76.068962$, $h(t) = 74$;
   3) $t = 74$, $h(t) = 65$;
   4) $t = 65$, $h(t) = 54$;
   5) $t = 54$, $h(t) = 420$;
   6) $t = 42$, $h(t) = 35$;
   7) $t = 35$, $h(t) = 30$;
   8) $t = 30$, $h(t) = 27$;
   9) $t = 27$, $h(t) = 27$;
   10) $t = 26$, $h(t) = 23$;
   11) $t = 23$, $h(t) = 12$;
   12) $t = 12$, $h(t) = 4$;

Since a failure deadline is found during this iteration, the optimal T_x' value is 10.5.

7. 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 timing deadlines when executed on a specified hardware platform. It is often the case that we need to know what is the minimum period of a task that will result in a system that is borderline schedulable.

In this paper, we proved that the minimum task period can be recalculated when a failure deadline is found during the QPA iteration. We addressed the problem when the task set's utilization is equal to 1, therefore, the implementation of sensitivity analysis can be improved by using different initial values of task period which are determined by Δ. There is no restriction on the task parameters, and there is no additional search or iterations required by the computations, hence the approaches are as efficient as QPA, and they can easily be incorporated into a design support tool.

As a future work, we will investigate the most suitable values of Δ by extensive simulations.

Acknowledgment

This work is supported by Natural Science Foundation Project of Chongqing (project No. CSTC,2010BB2010) and Science Research Foundation of Southwest University (project No. SWU110006).

References

[1] S.K. Baruah, A.K. Mok, and L.E. Rosier. Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor. *Proceedings 11th IEEE Real-Time System Symposium,* pp.182-190, 1990.

[2] E. Bini, M. Natale, and G. Buttazzo. Sensitivity Analysis for Fixed-Priority Real-Time Systems. *In Proceedings 18th Euromicro Conference on Real-Time Systems,* pp.13-22, 2006.

[3] M.L. Dertouzos. Control Robotics: The Procedural Control of Physical Processes. *Proceedings of the IFIP Congress,* pp.807-813, 1974.

[4] M. Harbour, J. Garcia, J. Gutierrez, and J. Moyano. MAST: Modeling and Analysis Suite for Real-Time Applications. *In Proceedings of Euromicro Conference on Real-Time Systems,* 2001.

[5] D. Katcher, H. Arakawa, and J. Strosnider. Engineering and Analysis of Fixed Priority Schedulers. *Software Engineering Journal*, 19(9):920-934, 1993.

[6] J. Lehoczky, L.Sha, and Y. Ding. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. *Proceedings of IEEE Real-Time Systems Symposium,* pp.166-171, 1989.

[7] J.Y.-T. Leung and M.L. Merrill. A Note on Preemptive Scheduling of Periodic, Real-Time Tasks. *Information Processing Letters,* pp.115-118, 1980.

[8] S. Punnekkat, R. Davis, and A.Burns. Sensitivity Analysis of Real-Time Task Sets. *In: ASIAN,* pp.72-82, 1997.

[9] J. Regehr. Scheduling Tasks with Mixed Preemption Relations for Robustness to Timing Faults. *In Proceedings IEEE Real-Time Systems Symposium,* 2002.

[10] I. Ripoll, A. Crespo, and A.K. Mok. Improvement in Feasibility Testing for Real-Time Tasks. *Journal of Real-Time Systems*, 11(1):19-39, 1996.

[11] M. Spuri. Analysis of Deadline Scheduled Real-time Systems. *Technical Report 2772, INRIA, France*, 1996.

[12] S. Vestal. Fixed-priority Sensitivity Analysis for Linear Compute Time Models. *IEEE Transactions on Software Engineering*, 20(4)1994.

[13] R. Yerraballi, R. Mukkamala, K. Maly, and H. Wahab. Issues in Schedulability Analysis of Real-Time Systems. *In Proceedings of 7th Euromicro Workshop on Real-Time Systems,* pp.87-92, 1993.

[14] F. Zhang and A. Burns. Schedulability Analysis for Real-Time Systems with EDF Scheduling. *Technical Report YCS-426-2008, Dept. of Computer Science, University of York, UK*, 2008.

[15] F. Zhang and A. Burns. Schedulability Analysis for Real-Time Systems with EDF Scheduling. *IEEE Transactions on Computers*, 58(9):1250-1258, 2009.

[16] F. Zhang, A. Burns, and S. Baruah. Sensitivity Analysis of Task Period for EDF Scheduled Arbitrary Deadline Real-Time Systems. *Proceedings 3rd IEEE International Conference on Computer Science and Information Technology,* (3), pp.23-28, 2010.

[17] F. Zhang, A. Burns, and S. Baruah. Sensitivity Analysis for EDF Scheduled Arbitrary Deadline Real-Time Systems. *Proceedings 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, in press,* 2010.