Improved Worst-Case Response-Time Calculations by Upper-Bound Conditions

Victor Pollex, Steffen Kollmann, Karsten Albers and Frank Slomka
Ulm University
Institute of Embedded Systems/Real-Time Systems
{firstname.lastname}@uni-ulm.de
http://www.uni-ulm.de/in/esys

Abstract

Fast real-time feasibility tests and analysis algorithms are necessary for a high acceptance of the formal techniques by industrial software engineers. This paper presents a possibility to reduce the computation time required to calculate the worst-case response time of a task in a fixed-priority task set with jitter by a considerable amount of time. The correctness of the approach is proven analytically and experimental comparisons with the currently fastest known tests show the improvement of the new method.

1. Introduction

Many embedded systems have real-time constraints. To determine whether these real-time constraints can be met or not, methods of the real-time analysis are used. During the last 30 years various approaches have been developed to verify the real-time behavior of embedded systems, particularly with regard to improving the run-time complexity of real-time and feasibility test algorithms. These test algorithms can be generally classified into exact tests and sufficient tests. The advantage of sufficient tests is that they deliver their results relatively fast compared to the exact tests, but a disadvantage is that certain task sets cannot be identified as feasible although they actually are. This acceptance problem of sufficient tests grows with an increasing utilization of the considered task set. Embedded system designers want to achieve high utilization of their embedded processors in order to minimize the final system costs. However, if fast feasibility tests reject suitable task sets, a careful design space exploration of the system is difficult.

A well known approach to exactly analyze the real-time behaviour of an embedded system is the response time analysis first introduced by Lehoczky [4]. Certain commercial tools like SymTA/S [7] are based on the response time analysis to evaluate distributed real-time systems by evaluating the end-to-end deadlines of chained tasks. Because of the iterative formulation of the response-time analysis, its run-time may increase in a non linear way. To counter this, it is necessary to further reduce the run-time complexity of the analysis.

In this paper we present such a reduction of run-time complexity. The approach is based on the idea of reducing the number of considered jobs by introducing an upper-bound condition to the sufficient real-time test. We will show that an algorithm based on this approach leads to a vastly improved performance of the test, especially when the utilization of the considered task set is rather high.

The paper is organized as follows: First we give an overview over the related work and introduce the model used in literature and in this paper. Chapter 3 presents the improvements we have explored. Both the theoretical background and the algorithm are given there. Experimental results showing the impact of the method to response time analysis are given in chapter 4 followed by a conclusion at the end.

2. Related Work

2.1. Overview

Many different approaches exist to determine the feasibility of a real-time system. Thereby, the approaches have to be as accurate as possible and as fast as possible. Much effort has been dedicated to reduce the run-time complexity of such analysis methods.

Lehoczky [4] has introduced the worst-case response time analysis with arbitrary deadlines. Sjödin and Hansson [6] have proposed some lower-bound conditions for the starting job of this test to improve the performance. A similar approach has been introduced by Bril, Verhaegh and Pol [3].

Liu and Layland [5] have developed a sufficient feasibility test for strict periodic task sets scheduled by the rate monotonic policy. Since then a lot of conditions for sufficient feasibility tests have been found. A good overview is given in [10]. Recently, Bini and Baruah [1] have presented an upper-bound condition for the worst-case response time of a task. By means of this bound condition a new sufficient feasibility analysis for periodic tasks has been introduced.

2.2. Run-Time Improved Response-Time Analysis

Before we introduce the new analysis method of the worst-case response time analysis, we will give an overview to the model we use, which is the same as in the related work.

We assume a real-time system which has n independent tasks $\Gamma = \{\tau_0, \dots, \tau_{n-1}\}$ scheduled on a single processor by a fixed-priority scheduling algorithm. Each task is characterized by a period T_i, which is the nominal time between two consecutive jobs in the absence of jitter, a worst-case execution time C_i of a task, which denotes a maximum amount of time that is needed on a reference processor for a job of the task to complete, a relative deadline D_i denoting the time after the arrival of a job, when the job has to be completed, a jitter J_i describing an interval in which the arrival of a job can vary. We will also use U_i to refer to the utilization of a task which equals to $U_i = \frac{C_i}{T_i}$. Further we will assume that $\sum_{i=0}^{n-1} U_i < 1$, because if $\sum_{i=0}^{n-1} U_i = 1$ then the bound introduced by (10) will not be usable for the proposed improvement under certain conditions. We will also assume that the tasks are sorted by decreasing priorities, meaning that τ_i has a higher priority than τ_j for all $i, j : 0 \le i < j < n$.

In a periodic static-priority task set with jitter

$$I_i^k = \min_{l \in \mathbb{N}}(I_{i,l}^k | I_{i,l}^k = (k+1) \cdot C_i + \sum_{j=0}^{i-1} \left\lceil \frac{I_{i,l-1}^k + J_j}{T_j} \right\rceil \cdot C_j) \tag{1}$$

with $I_{i,0}^k = 0$ is the completion-time of the k-th job of task τ_i as it was shown in [9]. Note that (1) is only valid for jobs within the level-i busy period as defined in [4].

The response time of a task is the difference between its completion time and its arrival time. $R_i^k = I_i^k - A_i^k$ where $A_i^k = \max(k \cdot T_i - J_i, 0)$. To calculate the worst case response time of task τ_i, currently the response times of all jobs within its busy period have to be evaluated [8]. The algorithm stops when the following condition holds $I_i^k \le A_i^{k+1}$, thus the worst-case response time is:

$$R_i = \max_{k \in 0 \dots m} R_i^k : m = \min(\{l : I_i^l \le A_i^{l+1}\}) \tag{2}$$

Sjödin and Hansson have presented in [6] several methods on how to reduce the run-time complexity of the response-time analysis. The main idea of their presented methods is to define lower-bound conditions where the analysis can start without changing the result of the analysis. This allows to skip every evaluation up to these lower-bound conditions.

Most of these methods reduce the number of iterations in the fix-point iteration by trying to start the fix-point iteration as close as possible to the final result. If the jobs of a task are evaluated in successive order, you can start at $I_{i,0}^k = I_i^{k-1} + C_i$ [6]. If the tasks are being evaluated in priority order, the calculation can start at $I_{i,0}^k = I_{i-1}^m + (k+1) \cdot C_i$.

Another method is to reduce the number of jobs to evaluate by starting with the last job that arrives at time 0 $t.u.$. For a detailed discussion of the explained methods see [6].

Bini and Baruah presented in [1] a sufficient feasibility test, in which they characterize the completion time as follows:

$$I_i^k = X_{i-1}((k+1) \cdot C_i) \tag{3}$$

where $X_i(h) = min_t\{t : H_i(t) \ge h\}$, with $H_i(t) = t - W_i(t)$ being the worst-case idle time and $W_i(t)$ being the worst-case workload of the i highest priority tasks over an interval of length t [1]. In 3.2 we will extend the equations to incorporate task jitter and derive a new upper-bound condition for the response time.

3. Advanced Job Reduction for Response-Time Analysis

To reduce the number of jobs which have to be considered, we propose two improvements. In section 3.1 we will first use the idea to start with the latest job for which we can show that the response time is greater or equal to the response time of any of the previous jobs. Second, with a much greater impact, we introduce a condition which allows us to stop evaluating jobs as early as possible. It is based on the upper-bound condition presented in [1] for a sufficient analysis by Bini and Baruah. We extend it to incorporate task jitter and use it to improve the performance of the exact analysis.

3.1. Removing Starting Jobs by Lower-Bound Condition

Sjödin and Hansson have shown in [6] that the evaluation can be started at job

$$h = \max_{k \in \mathbb{N}_0}\{k : A_i^k = 0\} \tag{4}$$

Lemma 1: The starting job as formulated in (4) is equal to $h = \lfloor \frac{J_i}{T_i} \rfloor$.

Proof: Assume $k_0 = \lfloor \frac{J_i}{T_i} \rfloor$. Because A_i^k is monotonically non-decreasing, we only have to show that $A_i^{k_0} = 0$ by showing that $k_0 \cdot T_i - J_i \leq 0$ and $A_i^{k_0+1} > 0$ by showing that $(k_0+1)T_i - J_i > 0$.

$$k_0 \cdot T_i - J_i = T_i(k_0 - \frac{J_i}{T_i}) \leq 0$$

$$(k_0+1)T_i - J_i = T_i(k_0 + 1 - \frac{J_i}{T_i}) > 0$$

□

We will now improve the lower-bound condition by following lemma:

Lemma 2: The response time of job $k = \lfloor \frac{J_i + C_i}{T_i} \rfloor$ of task τ_i is greater or equal to $max_{l=0..k-1} R_i^l$.

Proof: Assume $k_0 = \lfloor \frac{J_i + C_i}{T_i} \rfloor$. First we will show that

$$\begin{aligned} k_0 &\leq \frac{J_i + C_i}{T_i} \\ k_0 \cdot T_i - J_i &\leq C_i \\ A_i^{k_0} &\leq C_i \end{aligned} \tag{5}$$

Because of the nature of A_i^k we know that $A_i^{k_0-1} = 0$ and $A_i^{k_0+1} > C_i$. If $A_i^{k_0} = 0$ then k_0 would be the solution to (4), where $R_i^{k_0} > \max_{k \in 0 \ldots k_0 - 1} R_i^k$ follows. Now we will consider the case where $0 < A_i^{k_0} \leq C_i$. According to [6], the completion time of this job is by at least C_i greater compared to the completion time of the previous job

$$I_i^k \geq I_i^{k-1} + C_i \tag{6}$$

With (5) and (6) we can formulate the following equation

$$R_i^{k_0} - R_i^{k_0-1} = I_i^{k_0} - A_i^{k_0} - I_i^{k_0-1} \geq C_i - A_i^{k_0} \geq 0 \tag{7}$$

where $R_i^{k_0} \geq \max_{k \in 0 \ldots k_0 - 1} R_i^k$ follows.

□

Since only up to one job per task can be reduced, the new lower bound condition is just a small improvement compared to previous methods [6]. In Fig. 1 the new approach is formulated as an algorithm.

3.2. Removing Remaining Jobs by Upper-Bound Condition

After improving the lower-bound condition of the starting job of the tasks, we will now derive a new condition which will allow us to skip the evaluation of all remaining jobs from the point where the new condition holds. The new approach is based on the upper-bound condition for the response time given in

```
$I_{-1}^{\#0} = q = 0$
for $i = 0$ to $n-1$ /* n = number of tasks */
    $R_i^{max} = 0$
    $k = \lfloor \frac{J_i + C_i}{T_i} \rfloor$
    $I_i^{\#0} = I_{i-1}^{\#q} + kC_i$
    do
        $I_i^{\#0} = I_i^{\#q} + C_i$
        $q = 0$
        do
            $I_i^{\#q+1} = (k+1)C_i + \sum_{j=0}^{i-1} (\lceil \frac{I_i^{\#q} + J_j}{T_j} \rceil \cdot C_j)$
            $q = q + 1$
        while ($I_i^{\#q} > I_i^{\#q-1}$)
        $R_i^{max} = \max(I_i^{\#q} - A_i^k, R_i^{max})$
        $k = k + 1$
    while ($I_i^{\#q} > A_i^k$)
end
```

Figure 1. Algorithm by Sjödin and Hansson with the starting job improvement given in section 3.1

[1]. This upper-bound condition is adapted to incorporate task jitter by extending the upper-bound condition of the workload. The upper-bound condition of the workload is a linear function given by $w_j^o(t) = U_j t + x$. To calculate the upper-bound condition for a given task it is necessary to compute the constant x, which is performed by solving the equation $w_j^o(kT_j - J_j + C_j) = (k+1) \cdot C_j$ giving us the following value for $x = J_j \cdot U_j + C_j(1 - U_j)$, which results in following equation for the upper-bound condition of the workload

$$w_j^o(t) = U_j \cdot t + J_j \cdot U_j + C_j(1 - U_j) \tag{8}$$

Transforming (8) as in [1]

$$W_i^{\text{ub}}(t) = \sum_{j=0}^{i} (U_j t + J_j U_j + C_j(1 - U_j))$$

$$H_i^{\text{lb}}(t) = t \left(1 - \sum_{j=0}^{i} U_j \right) - \sum_{l=0}^{i} (J_l U_l + C_l(1 - U_l))$$

$$X_i^{\text{ub}}(h) = \frac{h + \sum\limits_{j=0}^{i} (J_j U_j + C_j(1 - U_j))}{1 - \sum\limits_{j=0}^{i} U_j}$$

where $W_i^{\text{ub}}(t)$ is the upper bound of the workload $W_i(t)$, $H_i^{\text{lb}}(t)$ the lower bound of the idle time $H_i(t)$ and $X_i^{\text{ub}}(h)$ the upper bound of $X_i(h)$ as mentioned in 2.2, leads to following upper-bound condition for the completion time

$$\iota_i^k = \frac{(k+1)C_i + \sum\limits_{j=0}^{i-1} (J_j U_j + C_j(1 - U_j))}{1 - \sum\limits_{j=0}^{i-1} U_j} \tag{9}$$

therefore the response time is bounded by

$$\rho_i^k = \iota_i^k - A_i^k \tag{10}$$

In the following sections of the paper it is assumed that $k_0 = \lfloor \frac{J_i}{T_i} + \frac{U_i}{1-\sum_{j=0}^{i-1} U_j} \rfloor$. We will now show that the response time is bounded from above by

$$R_i^{\text{ub}} = \frac{(k_0+1)C_i + \sum\limits_{j=0}^{i-1}(J_jU_j + C_j(1-U_j))}{1 - \sum\limits_{j=0}^{i-1} U_j} \tag{11}$$

by formulating the following lemma

Lemma 3: ρ_i^k has its maximum at $k = k_0$.

Proof: Assume $k \in \mathbb{N}_0$. First we will show that $A_i^{k_0+1} > 0$ by showing $k_0 > \frac{J_i}{T_i} - 1$ and $A_i^{k_0-1} = 0$ by showing $k_0 \leq \frac{J_i}{T_i} + 1$.

$$k_0 = \left\lfloor \frac{J_i}{T_i} + \frac{U_i}{1-\sum_{j=0}^{i-1} U_j} \right\rfloor \geq \left\lfloor \frac{J_i}{T_i} \right\rfloor > \frac{J_i}{T_i} - 1 \tag{12}$$

$$k_0 = \left\lfloor \frac{J_i}{T_i} + \frac{U_i}{1-\sum_{j=0}^{i-1} U_j} \right\rfloor \leq \left\lfloor \frac{J_i}{T_i} + 1 \right\rfloor \leq \frac{J_i}{T_i} + 1 \tag{13}$$

From (12) and (13) we can see that ρ_i^k is monotonically non-increasing for all $k > k_0$ and monotonically non-decreasing for all $k < k_0$. Because A_i^k has the property of being monotonically nondecreasing, we now know that $A_i^k > 0$ for all $k > k_0$ and that $A_i^k = 0$ for all $k < k_0$, thus

$$\rho_i^{k+1} - \rho_i^k = \frac{C_i}{1-\sum_{j=0}^{i-1} U_j} - T_i < 0 \;\forall k > k_0 \tag{14}$$

$$\rho_i^k - \rho_i^{k-1} = \frac{C_i}{1-\sum_{j=0}^{i-1} U_j} > 0 \;\forall k < k_0 \tag{15}$$

Now we have to show that $\rho_i^{k_0} - \rho_i^{k_0-1} \geq 0$ where it is assumed that $A_i^{k_0} > 0$, because in the case that $A_i^{k_0} = 0$ the result is the same as in (15)

$$\begin{aligned}\rho_i^{k_0} - \rho_i^{k_0-1} &= \frac{C_i}{1-\sum_{j=0}^{i-1} U_j} - k_0T_i + J_i \\ &= T_i(\frac{J_i}{T_i} + \frac{U_i}{1-\sum_{j=0}^{i-1} U_j} - k_0) \geq 0\end{aligned}$$

and additionally we have to show that $\rho_i^{k_0+1} - \rho_i^{k_0} \leq 0$. Now we will assume that $A_i^{k_0} = 0$ otherwise it would be like (14)

$$\begin{aligned}\rho_i^{k_0+1} - \rho_i^{k_0} &= \frac{C_i}{1-\sum_{j=0}^{i-1} U_j} - (k_0+1)T_i + J_i \\ &= T_i(\frac{J_i}{T_i} + \frac{U_i}{1-\sum_{j=0}^{i-1} U_j} - (k_0+1)) < 0\end{aligned}$$

□

Directly from lemma 3 we can formulate the following sufficient feasibility test:

Corollary 1: A task set scheduled by static priorities is feasibly if for all tasks the upper-bound condition of the response time is less or equal to the deadline

$$\forall i \; R_i^{\text{ub}} = \frac{(k_0+1)C_i + \sum\limits_{j=0}^{i-1} J_jU_j + C_j(1-U_j)}{1 - \sum\limits_{j=0}^{i-1} U_j} \leq D_i \tag{16}$$

Note that in the case that all tasks do not have a jitter $\forall i : J_i = 0$, this sufficient test is equivalent to the test presented in [1].

Now we can finally formulate the condition which will help us reduce the number of jobs which have to be considered by following corollary

Corollary 2: If $\max\limits_{l \in 0 \ldots k} R_i^l \geq \rho_i^{k+1}$ then $R_i = \max\limits_{l \in 0 \ldots k} R_i^l$.

Proof: In the case that $k < k_0 - 1$ the condition $\max_{l \in 0 \ldots k} R_i^l \geq \rho_i^{k+1}$ does not hold, hence only the case where $k \geq k_0 - 1$ has to be considered. From lemma 3 we know that at $k = k_0$, ρ_i^k has its maximum and for any successive job ρ_i^k does not increase, meaning that at ρ_i^k for any $k \geq k_0$ it is the maximum response time of all the following jobs. Now if the response time of all following jobs is less or equal to the currently computed maximum response time, the currently computed maximum response time is the actual worst-case response-time of the analyzed task. □

To implement this, simply change line 15 of Fig. 1 from "while $(I_i^{\#q} > A_i^k)$" to "while $(I_i^{\#q} > A_i^k) \wedge (\rho_i^k > R_i^{max})$"

From corollary 2 we can also formulate an exact feasibility test for a task

Corollary 3: If the currently analyzed job $k \geq k_0 - 1$ and the currently computed maximum response time $\max_{l \in 0 \ldots k} R_i^l \leq D_i$ and the maximum response time of all following jobs $\rho_i^{k+1} \leq D_i$ then τ_i is feasible.
See Fig. 2 for an example on how to incorporate this into an exact feasibility test for a task set.

4. Experiments

The new worst-case response time analysis and exact feasibility test as developed in section 3 is compared with the work of Sjödin and Hansson [6]. For this comparison we have generated random task sets with periods distributed uniformly ranging from 10 *t.u.* up to 10,000,000 *t.u.*, jitter was also generated by uniformly distributing it in the range between 0 and 5 times its period, $0 \leq J_i < 5 \cdot T_i$, utilization was distributed with the UUniFast algorithm as proposed in [2]. With the generated utilization, the worst-case

```
$I^{\#0}_{-1} = q = 0$
for $i = 0$ to $n-1$ /* $n$ = number of tasks */
    $k = \lfloor \frac{J_i + C_i}{T_i} \rfloor$
    $I^{\#0}_{i} = I^{\#q}_{i-1} + kC_i$
    do
        $I^{\#0}_{i} = I^{\#q}_{i} + C_i$
        $q = 0$
        do
            $I^{\#q+1}_{i} = (k+1)C_i + \sum_{j=0}^{i-1} (\lceil \frac{I^{\#q}_{i} + J_j}{T_j} \rceil) \cdot C_j$
            $q = q + 1$
            if $(I^{\#q}_{i} - A^k_i > D_i)$
                return "not␣feasible"
        while $(I^{\#q}_{i} > I^{\#q-1}_{i})$
        $k = k + 1$
    while $(I^{\#q}_{i} > A^k_i) \wedge (\rho^{\mathbf{k}}_{\mathbf{i}} > \mathbf{D_i})$
end
return "feasible"
```

Figure 2. An exact feasibility test

execution time was set to $C_i = U_i \cdot T_i$. Finally, deadlines were set to twice the period $D_i = 2 \cdot T_i$.

Experiments conducted with the starting job as shown in 3.1 resulted in overall slower computation times compared to starting at the job proposed by Sjödin and Hansson in [6]. We assume this is due to the uncertainty of its arrival time. If the evaluation starts at $k = \lfloor \frac{J_i + C_i}{T_i} \rfloor$ the value of A^k_i has to be computed. If the evaluation starts at $k = \lfloor \frac{J_i}{T_i} \rfloor$ then $A^k_i = 0$ and does not have to be computed. The proposed new starting job does save us to evaluate some jobs, but the additional computation time needed to compute the arrival-time outweighs the gains from the saved jobs. Because of this we started to evaluate at $k = \lfloor \frac{J_i}{T_i} \rfloor$ in our experiments.

First we sampled the utilization axis in steps of 0.1% starting at 0.1% up to 99.9%. For each step 1000 random task sets were generated with 100 tasks each.

Fig. 3 and 4 are showing the average run-time and the average number of jobs evaluated per task set each as a function of the utilization of a task set. Note that the y-axis is scaled logarithmically. As can be seen the higher the utilization the more efficient the condition becomes. In this experiment the improvement of the run-time over all evaluated task sets amounts to almost 50%. If we only consider task sets with an utilization greater or equal to 0.9 then the improvement rises to almost 66%.

In additional experiments the utilization is set to 0.8 and the number of tasks was varied from 5 up to 1000 tasks per task set in increments of 1 task. For each sample we generated 100 random task sets.

Fig. 5 and 6 are again showing the average run-time and the average number of jobs evaluated per task set this time as a function of the number of tasks in a task set. The experiments give the result that with an increasing number of tasks in a task set, our new approach is more efficient than the previous one presented by Sjödin and Hansson.

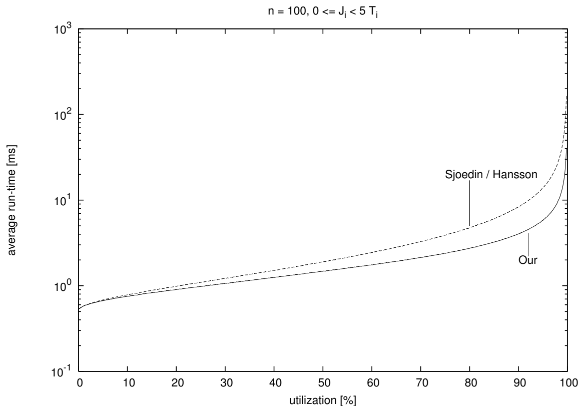

Figure 3. Run-time vs. utilization

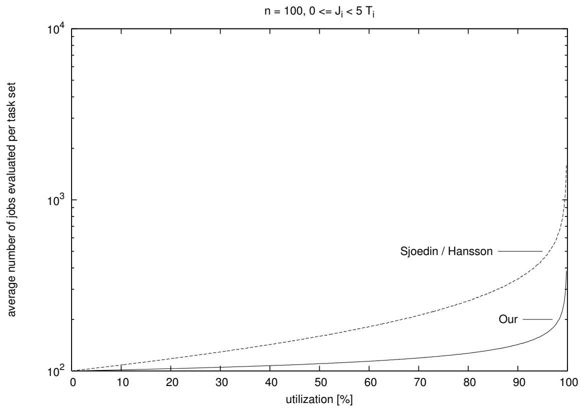

Figure 4. Jobs vs. utilization

Fig. 7 and 8 show the same relationships as Fig. 5 and 6 with the difference that the exact feasibility test algorithms were used, see also Fig. 2.

5. Conclusion

In this paper the improvements of [1] and [6] are combined and extended to formulate a new faster analysis algorithm for the response-time analysis. The extension of the upper-bound condition for the worst-case response time given to the response-time analysis to consider jitter and exploiting its properties, improves the response-time analysis dramatically. This result is given because the new algorithm formulated in this paper reduces the number of jobs considered by the analysis. The improvement is compared experimentally with the work of Sjödin and Hansson. The experiments show that by using the new algorithm, larger task sets

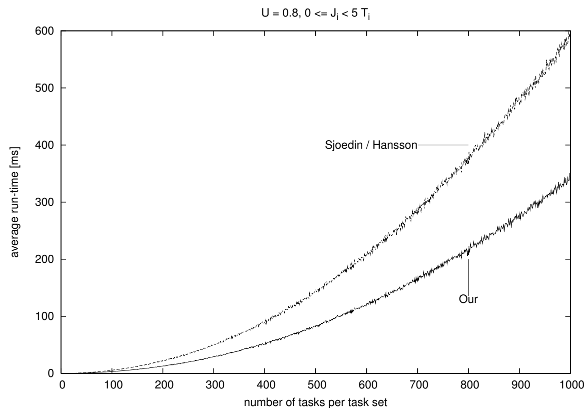

Figure 5. Run-time vs. number of tasks

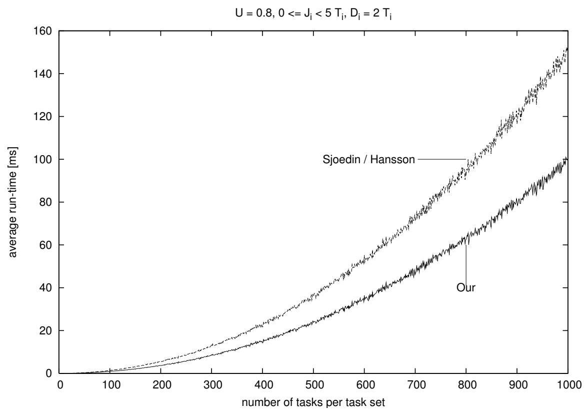

Figure 7. Run-time vs. number of tasks (feasibility test)

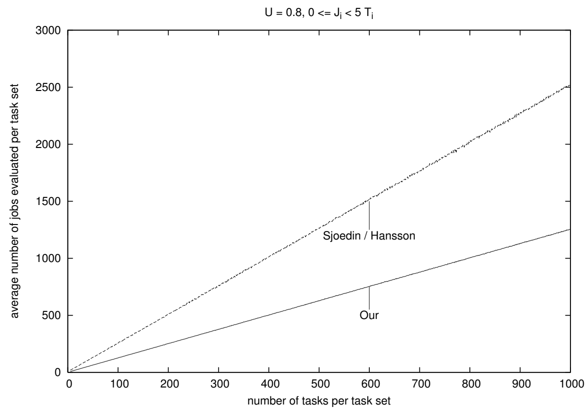

Figure 6. Jobs vs. number of tasks

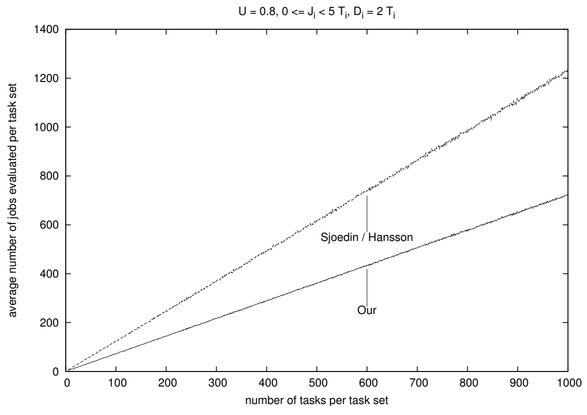

Figure 8. Jobs vs. number of tasks (feasibility test)

and task sets with a rather high utilization are evaluated faster. In future work we will explore the effect of the run-time improvement in the context of distributed systems.

References

[1] Enrico Bini and Sanjoy K. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. In *Proceedings of the 15th International Conference on Real-Time and Network Systems*, pages 95–104, March 2007.

[2] Enrico Bini and Giorgio C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[3] Reinder J. Bril, Wim F. J. Verhaegh, and Evert-Jan D. Pol. Initial values for online response time calculations. In *Proceedings of 15th Euromicro Conference on Real-Time Systems*, pages 13–22, 2003.

[4] John P Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, December 1990.

[5] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[6] Mikael Sjödin and Hans Hansson. Improved response-time analysis calculations. In *IEEE Real-Time Systems Symposium*, pages 399–408, 1998.

[7] Symtavision. http://www.symtavision.com.

[8] K. W. Tindell. An extendible approach for analysing fixed priority hard real-time tasks. *The Journal of Real-Time Systems*, 6:133–151, 1994.

[9] Ken Tindell and John Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40:117–134, April 1994.

[10] Jianjia Wu, Jyh-Charn Liu, and Wei Zhao. On schedulability bounds of static priority schedulers. In *RTAS '05: Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pages 529–540, Washington, DC, USA, 2005. IEEE Computer Society.