# Improved Analysis for Real-Time Tasks With Offsets

## Advanced Model

Jukka Mäki-Turja
jukka.maki-turja@mdh.se

Mikael Sjödin
mikael.sjodin@mdh.se

Mälardalen Real-Time Research Centre

MRTC Report nr 101

May 2003

## Abstract

*We present an improvement to the approximative response-time analysis for fixed-priority scheduled tasks with offsets presented by Tindell [Tin92] and Palencia Gutierrez et al. [GH98].*

*The improvement tightens the analysis (i.e. makes it less pessimistic) by removing unnecessary overestimation of the interference a task can impose on other tasks. This overestimation does not cause any pessimism in response-time analysis without offsets (neither in the exact analysis with offsets), but as we show, in the approximative offset analysis it causes significant pessimism.*

*We present an evaluation (by simulation) which show that, compared to the previous method, our method can calculate about 10% lower response-times for up to 70% of the tasks.*

*In this paper we first present and discuss our method using a simplified task model and later in the paper we show how our method applies to the more advanced model of Palencia Gutierrez [GH98].*

# 1 Introduction

When designing hard real-time systems, methods to guarantee that there is no deadline violated are needed. One important class of such analysis techniques is schedulability analysis. A schedulablilty analysis is performed with a set of tasks and their resource demands as input. The output is a statement whether or not it is guaranteed that all deadlines will be met.

A powerful and well established schedulability analysis technique is the *Response-Time Analysis* (RTA) [JP86, ABT+93, ABD+95]. RTA is applicable to systems where tasks are scheduled in strict priority order. Priority scheduling is the predominant scheduling technique used in real-time operating systems today.

Traditionally RTA is based on the *critical instant* assumption of Liu and Layland [LL73]. In their work they assume that tasks are independent, and hence the worst possible phasing of tasks is when all tasks are released at the same instant in time. However, in most real-time systems not all tasks are independent. In fact, it is quite common with precedence relations and/or timing offset between subsets of tasks. One way to organise such dependencies is to group subset of tasks with dependencies into *transactions*. Tindell proposed an extension to the RTA that take timing offsets between tasks into account [Tin92] (a timing offset tells how much time should elapse between the release of two tasks). Tindell provided an exact algorithm for calculating response time for tasks with offsets. However, this algorithm becomes computationally intractable for anything but small task sets due to its exponential time complexity. In order to deal with this problem, Tindell also provided an approximation algorithm that is polynomial in time and give pessimistic but *safe*[1] results.

Palencia Gutierrez *et al.* showed how precedence relations can be modelled by timed offsets [GH98]. Hence this paper only considers timed offsets. Another use for timed offsets is to model and analyse systems with a dual scheduling discipline; providing both static cyclic scheduling and dynamic scheduling. We have earlier discussed the industrial relevance and trends for such dual scheduling systems and shown how offset relations can be used for schedulability analysis for such systems [MTS02].

In this paper we will present an improvement to the approximative offset analysis given by Tindell [Tin92] and Palencia Gutierrez *et al.* [GH98]. The main contributions of this paper are:

1. Introduction of an tight approximative method to calculate response-times that are often lower than, and never greater than, those calculated by the existing approximative methods.

[1] In the context of scheduling analysis, safe means overestimation.

2. Evaluation by simulation of the method and how it performs compared to the original approximation and exact analysis.

The above methods are not dependent on each other and can hence be deployed independently in a tool for response-time calculations. For pedagogical reasons and to distinguish our contributions from related work, we will in the main paper use a simplified system model (see section 2) both for describing existing frameworks and our improvements. However, in Appendix A we also present our analysis in the context of the more advanced model used by Palencia Gutierrez *et al.* [GH98].

**Paper Outline:** In section 2 we outline related work and describe our system model and assumptions. In section 3 we continue by recapitulating the existing approximative methods. We will then in section 4 present our contribution to calculate tighter response times. Section 5 presents our evaluation, and finally in section 6, we draw conclusions and summarise.

# 2 Setting the Scene

In this section we present related work and the basic system model.

## 2.1 Background

Tindell proposed an extension to the Response-Time Analysis (RTA) which takes timing offsets between tasks into account [Tin92]. He modifies the notion of the critical instant and limits the pessimism of the original Liu and Layland's RTA when tasks have offset relations. His modified critical instant is based on the observation that not all task in a transaction can be released at the same time. His conclusion is that only one task from each transaction can be released at the critical instant.

Several researchers have extended the work provided by Tindell. In this paper we focus on the approximative analysis, and we will here provide an overview of work in that area. Palencia Gutierrez *et al.* generalised, formalised and extended the work by Tindell [GH98]. They introduced dynamic offsets, allowed offsets and deadlines larger than period, and made a slight improvement of the approximation algorithm. Palencia Gutierrez *et al.* also provided improvements in order to calculate tighter response-times in certain situations [GH99]. Redell further improved their work by giving a method to calculate even lower response-times [Red03].

However, both improvements [GH99, Red03] are only useful in very special circumstances where task priorities are chosen in a particular way and task jitter

is extremely high.[2] Hence, their improvements are of limited generality. The focus of their methods is on finding infeasible execution orders between tasks and removing these execution orders from the set of possible critical instants. The method we present in this paper is more general and can straight forwardly be combined with the above described improvements.

## 2.2 System Model

The system model used is as follows: The task set, $\Gamma = \{\Gamma_1, \ldots, \Gamma_k\}$, consists of a number of transactions $\Gamma_i$. A transaction, $\Gamma_i$, is defined by a set of tasks $\tau_{i1}, \ldots, \tau_{in}$ and a period $T_i$:

$$\Gamma_i := \langle \{\tau_{i1}, \ldots, \tau_{in}\}, T_i \rangle$$

The period, $T_i$, is used as a minimum interarrival time. Hence, transactions need not be periodic as long as its interarrival time is bounded by $T_i$. A task, $\tau_{ij}$, is defined by a worst case execution time ($C_{ij}$), an offset ($O_{ij}$), a deadline ($D_{ij}$), and a priority ($P_{ij}$), as follows:

$$\tau_{ij} := \langle C_{ij}, O_{ij}, D_{ij}, P_{ij} \rangle$$

The offset denotes the earliest release time of the task relative to the start of the transaction. The first subscript denotes which transaction the task belongs to, and the second subscript denotes the number of the task within the transaction. (No ordering of the tasks within a transaction is assumed, and tasks within a transaction are allowed to overlap in time.)

As stated in the introduction, our objective in this section of the paper is to present our improvements in a simple model making our contribution clear. Therefore, we introduce some simplifying assumptions:

- Offsets less than period, i.e. $O_{ij} \leq T_{ij}$.
- Deadline less than period, i.e. $D_{ij} < T_i$.
- Jitter for periodic transactions/tasks is not modelled.
- Blocking on shared resources (e.g. using semaphores) is not modelled.
- Unique priority for each task is assumed, i.e. $P_{ij} \neq P_{kl}$.

In Appendix A we present, using the model and method of Palencia Gutierrez [GH98], our tighter analysis without most of these simplifying assumptions.

[2] Priority needs to be chosen so that transactions can "interlock" each other, and the jitter needs to be in parity with, or greater than, the task's periods. Otherwise the proposed improvements will have little or no effect.

# 3 Existing Offset Analysis

In this section we restate the approximative analysis for tasks with timing offset presented by Tindell [Tin92] and Palencia Gutierrez *et al.* [GH98]. We use the simplified system model of section 2.2. We also present some figures to show the intuition behind the original offsets analysis.

When analysing tasks with offsets, exactly one task for each transaction is assumed to coincide with the critical instant [Tin92]. For instance, consider the transaction depicted in figure 1. It consists of two tasks, $\tau_{i1}$ and $\tau_{i2}$, and is defined as:

$$C_{i1} = 2 \quad O_{i1} = 0 \quad C_{i2} = 4 \quad O_{i2} = 4 \quad T_i = 12$$

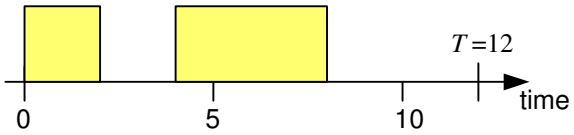


Figure 1: An example transaction with two tasks

In this example there are two candidate tasks to coincide with the critical instant. Figures 2(a) and 2(b) shows how interference from this transaction grows over time, as *stepped stair* functions, for each of the candidates. (Since the transaction is periodic, the depicted patterns repeat themselves after one period.)

The behaviour visualised in figure 2(a) or 2(b) is formally expressed as follows: For each candidate task, $\tau_{ic}$, in transaction $\Gamma_i$, that could coincide with the critical instant, we can calculate the amount of interference $\Gamma_i$ poses on the *task under analysis*, $\tau_{xy}$, during a time interval of length $t$. We call that interference $I(\tau_{xy}, \tau_{ic}, t)$ and is defined as:

$$\begin{aligned} I(\tau_{xy}, \tau_{ic}, t) &= \sum_{\tau_{ij} \in hp_i(\tau_{xy})} \left\lceil \frac{t'}{T_i} \right\rceil C_{ij} \\ t' &= t - phase(\tau_{ic}, \tau_{ij}) \end{aligned} \tag{1}$$

where $hp_i(\tau_{xy})$ represents the set of tasks belonging to transaction $\Gamma_i$ with priority greater than to the priority of $\tau_{xy}$. $\lceil\ \rceil$ denotes the ceiling function (returning the smallest integer that is equal to or greater than its operand), and $phase(\tau_{ic}, \tau_{ij})$ describes the distance in time from the release of $\tau_{ic}$ to the release of $\tau_{ij}$, defined as:[3]

$$phase(\tau_{ic}, \tau_{ij}) = O_{ij} - O_{ic} \mod T_i \tag{2}$$

[3]Note that $0 \leq phase(\tau_{ic}, \tau_{ij}) < T_i$.

In equation 1 $t'$ represents the time during which $\tau_{ij}$ has had a chance to interfere with $\tau_{xy}$. Or, informally, $t'$ "starts ticking" once $t$ has reached the offset $O_{ij}$ ($\tau_{ic}$ is released at $t = 0$).[4]

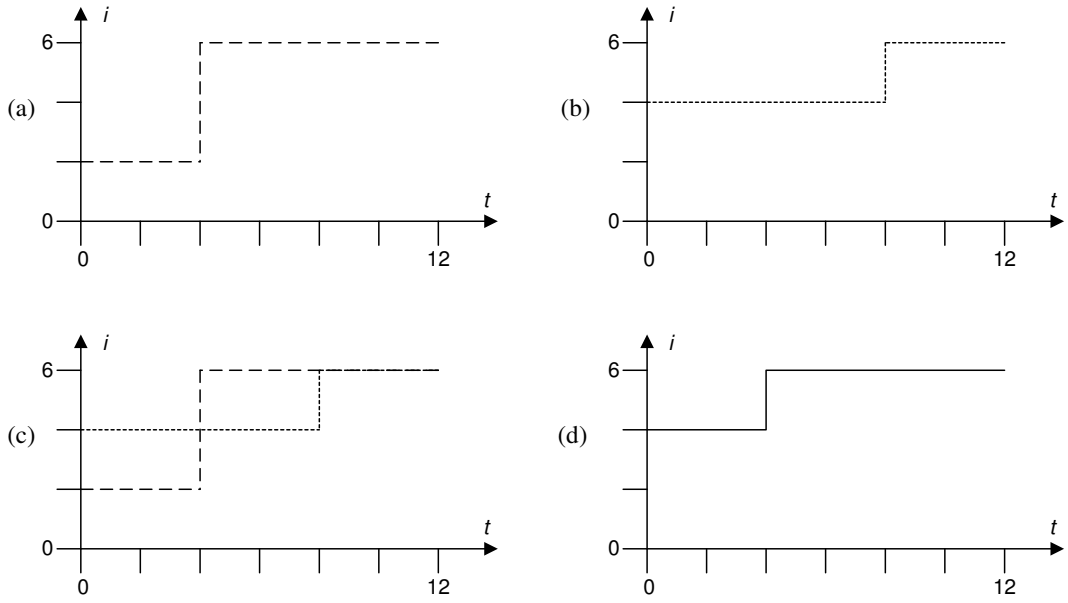


Figure 2: Interference caused by our example transaction

Since we beforehand cannot know which task in each transaction coincides with the critical instant, the exact analysis tries every possible combination. However, since this is computationally intractable for anything but small task sets the approximative analysis [Tin92, GH98], defines a (safe) approximation of the interference caused by transaction $\Gamma_i$:

$$A(\tau_{xy}, \Gamma_i, t) = \max_{\tau_{ic} \in \Gamma_i} I(\tau_{xy}, \tau_{ic}, t) \tag{3}$$

Figure 2(c) shows both candidate tasks' interference ovelayed and figure 2(d) shows the resulting approximation expressed by equation 3.

Given the definition of the interference the response time for our task under analysis, $R_{xy}$, is:

$$R_{xy} = C_{xy} + \sum_{i \in \Gamma} A(\tau_{xy}, \Gamma_i, R_{xy}) \tag{4}$$

which is solved by fix-point iteration, starting with $R_{xy} = 0$.

[4]Note that $\forall t : t' > -T_i$, hence the ceiling expression in equation 1 is never negative.

# 4 Tighter Offset Analysis

The approximation algorithm presented in the previous section can be tightened, i.e. there is a potential to lower the calculated response-times. Consider again the transaction $\Gamma_i$ depicted in figure 1 on page 5. For a lower priority task $\tau_{xy}$ with $C_{xy} = 2$ the fix-point iterative response-time calculation of equation 4 is as follows:[5]

| Iter# | $t$ | $I(\tau_{i1})$ | $I(\tau_{i2})$ | A() | $R_{xy}$ |
|---|---|---|---|---|---|
| 0 | | | | | 0 |
| 1 | 0 | 0 | 0 | 0 | 2 |
| 2 | 2 | 2 | 4 | 4 | 6 |
| 3 | 6 | 6 | 4 | 6 | 8 |
| 4 | 8 | 6 | 4 | 6 | 8 |

Where column "Iter#" denotes the iteration number, "$t$" the time interval, "I($\tau_{i1}$)" and "$I(\tau_{i2})$" denotes $I(\tau_{xy}, \tau_{ic}, t)$ for the two candidate tasks, "A()" the value of $A(\tau_{xy}, \Gamma_i, t)$, and "$R_{xy}$" the calculated response-time for the iteration.

The calculated response time is $R_{xy} = 8$. However, it can easily be seen that a task with $C_{xy} = 2$ can never be preempted by both tasks $\tau_{i1}$ and $\tau_{i2}$ (since both tasks are separated by at least 2 units of idle time). Hence the actual worst case response-time is $R_{xy} = 6$.

One property of the ceiling expression of equation 1 on page 5 is that $I(\tau_{xy}, \tau_{ic}, t)$ does not return the interference experienced by $\tau_{xy}$ until time $t$, instead it returns the amount of interference *released* for execution. If we modify equation 1 in order to return the interference *experienced* by $\tau_{xy}$ we get a *slanted stair* function. The two slanted stair functions for our example transaction are shown in figures 3(a) and 3(b).

The slanted stairs are obtained by modifying equation 1 so that the "last" task instance does not interfere with its full execution time (unless the interval $t$ is sufficiently large), rather a portion of it. Our redefined version of $I(\tau_{xy}, \tau_{ic}, t)$ is:

$$
\begin{aligned}
I(\tau_{xy}, \tau_{ic}, t) = & \sum_{\tau_{ij} \in hp_i(\tau_{xy}) \cap \Gamma_i} \left( \left( \left\lfloor \frac{t'}{T_i} \right\rfloor + 1 \right) C_{ij} - x \right) \\
t' = & t - phase(\tau_{ic}, \tau_{ij}) \\
x = & \begin{cases} 0 & t' < 0 \\ \max\left(0, C_{ij} - (t' \mod T_i)\right) & t' \geq 0 \end{cases}
\end{aligned}
\tag{5}
$$

where $phase(\tau_{ic}, \tau_{ij})$ is defined in equation 2 on page 5, $\lfloor\ \rfloor$ is the floor operator (returning the greatest integer that is equal to or lower than its operand), and $x$

[5] In this example no other higher priority tasks exists in the system.

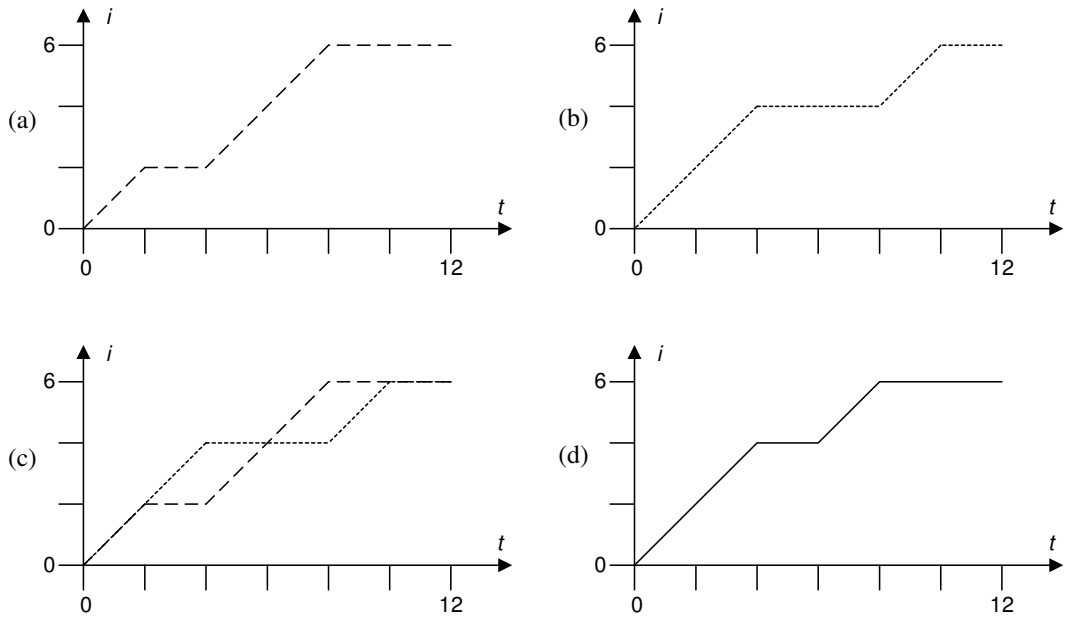


Figure 3: Interference *generated* by our example transaction

is used to generate the slants of the interference function. Figure 4(a) illustrates the a sequence of releases of a task $\tau_{ij}$, and figure 4(b) shows how the value of $x$ varies accordingly.

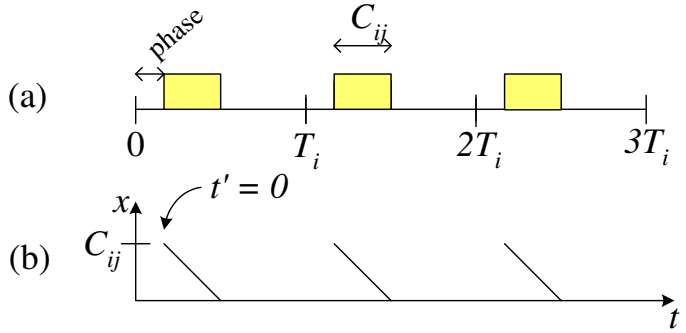


Figure 4: Relation between task release and $x$

The slanted stairs generated by equation 5 are shown in figures 3(a) and 3(b), and figure 3(c) shows them overlayed. Using our new version of $I(\tau_{xy}, \tau_{ic}, t)$ in equation 3 on page 6 we get the combined slanted stairs function shown in figure 3(d).

Combining equations 3, 4, and 5 we can now calculate a new response-time $R_{xy}$ as follows:

| Iter# | $t$ | $I(\tau_{i1})$ | $I(\tau_{i2})$ | A() | $R_{xy}$ |
|---|---|---|---|---|---|
| 0 | | | | | 0 |
| 1 | 0 | 0 | 0 | 0 | 2 |
| 2 | 2 | 2 | 2 | 2 | 4 |
| 3 | 4 | 2 | 4 | 4 | 6 |
| 4 | 6 | 4 | 4 | 4 | 6 |

Where column "Iter#" denotes the iteration number, "$t$" the time interval, "$I(\tau_{i1})$" and "$I(\tau_{i2})$" denotes $I(\tau_{xy}, \tau_{ic}, t)$ for the two candidate tasks, "A()" the value of $A(\tau_{xy}, \Gamma_i, t)$, and "$R_{xy}$" the calculated response-time for the iteration.

We note that our new definition of $I(\tau_{xy}, \tau_{ic}, t)$ make the analysis able to "see" the empty slot between tasks $\tau_{i1}$ and $\tau_{i2}$ something the original analysis overlooked. Hence, the calculated response-time (6) is lower than that of the original analysis (8). Below we prove that our new interference function is never greater than the original interfere function. Hence, the analysis based on our new definition of $I(\tau_{xy}, \tau_{ic}, t)$ always yields lower or equal response-time than does the original analysis.

**Theorem:** For a given task under analysis, $\tau_{xy}$, and one candidate task, $\tau_{ic} \in \Gamma_i$, our new definition of $I(\tau_{xy}, \tau_{ic}, t)$ (equation 5) is never greater than the old definition (equation 1).

**Proof:** We prove this by noting that there are only two terms that differ between equations 1 and 5, and that none of them can contribute to making equation 5 greater than equation 1:

1. $x$ is used to decrease the calculated value. Since, $x$ is never negative it can never contribute to making equation 5 greater than equation 1.

2. The expression $\lceil \frac{t'}{T_i} \rceil$ is replaced by $\lfloor \frac{t'}{T_i} \rfloor + 1$. For this expression we have two cases:

   (a) $t'$ that is not a multiple of $T_i$:
   Since $\lceil a \rceil = \lfloor a \rfloor + 1$ for non-integers $a$, the floor+1 expression cannot contribute to making equation 5 greater than equation 1 for this case.

   (b) $t'$ that is a multiple of $T_i$:
   Since $\lceil a \rceil = \lfloor a \rfloor$ for integers $a$, the floor+1 expression will return 1 more then the ceiling expression. This will result in an increase in the final $I(\tau_{xy}, \tau_{ic}, t)$ of $C_{ij}$. However, for $t'$ that is a multiple of $T_i$, the expression $t' \mod T_i$ will return zero and the resulting $x$ is exactly $C_{ij}$, hence cancelling the increase caused by the floor+1 expression.□

Our new definition of $I(\tau_{xy}, \tau_{ic}, t)$ is still safe, i.e., we never underestimate the interference, since we never reduce any experienced interference. The intuition for this is that the actual experienced interference can never grow faster than the interval $t$, and since the slope of our slants is 1, our interference will grow exactly as fast as $t$.

## 4.1 Discussion

At first glance, it is not obvious that lowering the interference function $I(\tau_{xy}, \tau_{ic}, t)$ should automatically give lower response-time. In fact, the stepped-stair interference-function has been used for many years to represent the interference in RTA [ABT+93, ABD+95], without introducing any pessimism.

The reason stepped stairs (in analysis without offsets) does not introduce pessimism can be found in our previous work [SH98]. We showed that the fix-point iteration will terminate when the sum of all interference functions crosses the line from origin with slope 1. Hence, replacing stepped stairs with slanted stairs (with slope 1) will not contribute to earlier fix-point convergence.

However, in approximative response-time analysis with offsets, the interference functions are not used directly in the fix-point iteration. Instead they are subjected to a maximisation function (equation 3 on page 6). The maximisation function is used, for each $t$, to select the candidate scenario with the highest interference. By using slanted-stair functions as input to the maximisation function we essentially "delay" the time it takes for one low-interference scenario to overtake a high-interference scenario.

Figure 5(a) shows our example transaction with two arrows denoting the two possible scenarios for the critical instant (one "dashed" scenario and one "dotted" scenario). Figures 5(b) and 5(c) shows the stepped stairs and slanted stairs interference functions respectively for both scenarios. At times $t < t1$ the dotted scenario is the one with highest interference. The time $t1$ corresponds to the release of the second task in the dashed scenario. For the stepped stairs case, this means immediately adding another 4 units if interference to the dashed scenario, hence making it the scenario with the highest interference. However, for the slanted stairs case, the time $t1$ means that the dashed line starts to increase, but not until time $t2$ it catches up with the dotted scenario. Hence, the interval between $t1$ and $t2$ represent the time by which the slanted stairs "delay" the dashed scenario to catch up with the dotted scenario. If fix-point convergence can be achieved during this interval then our tighter analysis will calculate a lower response time than does the original analysis.

Our tighter analysis has here been presented using a simplistic system model. In section 2.2 we outlined the major simplifications compared to other work

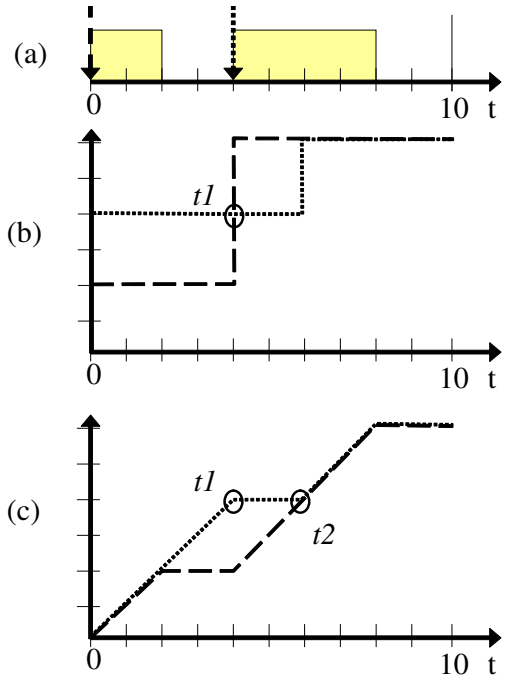


Figure 5: Stepped stairs vs. slanted stairs

[GH99, Red03]. None of these simplification affect the maximisation operation of the approximative analysis (equation 3), hence our technique of using slanted stairs is directly applicable to system models where these restrictions have been lifted. In Appendix A we present, using the model and method of Palencia Gutierrez *et al.* [GH98], our tighter analysis without most of these simplifying assumptions.

# 5 Evaluation

In order to evaluate the impact our proposed improvement we have implemented our improvement, the original approximative analysis [Tin92, GH98], and the original exact analysis [Tin92]. Using these implementation and a synthetic task-generator we have performed a simulation evaluation.

## 5.1 Description of Simulation

In our simulator we generate task sets that are used as input to the different analysis algorithms. The task generator takes the following parameters:

- Desired systems load (in % if total CPU utilisation),
- the number of transactions to generate, and
- the number of tasks per transaction.

Using these parameters a task set with the following properties is generated (figure 6 shows an example task-set generated by our task generator).:

- Transaction periods ($T_i$) are randomly distributed in the range 1.000 to 1.000.000 (uniform distribution).
- Each offset ($O_{ij}$) is randomly distributed within the transaction period (uniform distribution).
- The total system load is proportionally distributed over all transactions.
- The execution times ($C_{ij}$) are chosen as a fraction of the time between two consecutive offsets in the transaction. The fraction is the same throughout one transaction. The fraction is selected so that the the transaction load (above) is obtained.
- The priorities as assigned in rate monotonic order [LL73].

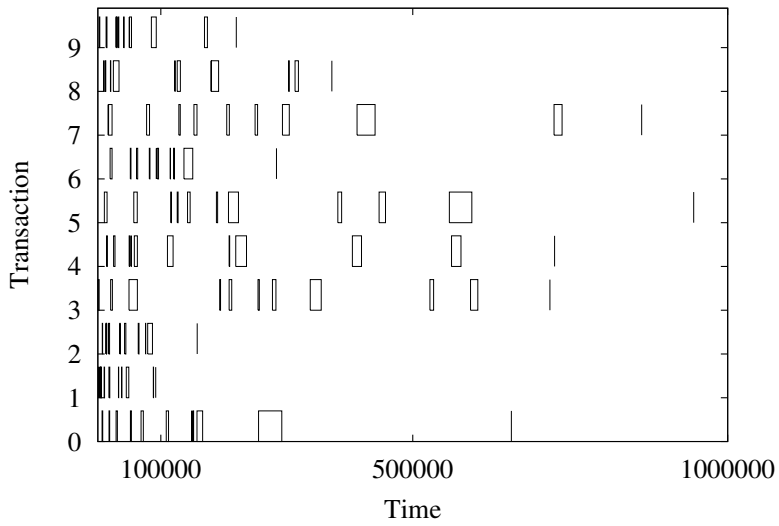


Figure 6: An example task set. Load=90%, Transactions=10, Tasks/Trans.=10

The results in section 5.2 has been obtained by taking the mean values of 100 simulated task-sets for each point in each graph. The graphs also show the 95%

confidence interval for the mean values.[6] We have measured two metrics from the simulations:

1. "Task with improvements (%)" — This metric measures the fraction of tasks that has a lower response time compared to the original approximative analysis. Note, that for this metric the original approximative analysis is used as a base line, hence no curve is plotted for that method. Also, note that this metric says nothing about the size of the difference in response times.

2. "Average improvement (%)" — For the tasks where there was an improvement using the exact analysis, or our tighter analysis, compared to the original approximative analysis, we calculated the average decrease in response time in percent. Similar to the previous metric, the original approximative analysis is used as a base line, hence no curve is plotted for that method.

## 5.2 Simulation Results

In the simulations we have varied our three task-generator parameters in different ways. Figure 7 on the next page shows a subset of the simulation results. Note, the exact analysis can only be run on small task sets, hence it is not present in figures with large tasks sets.

In figures 7(a) to 7(d) we compare the calculated response-times of the exact analysis (Exact), the original approximative analysis (baseline) and our tighter approximative analysis (Tight). The figures show that our tighter method follows the exact analysis quite close (i.e., our tighter analysis is almost as good as the exact analysis), both with respect to the number of tasks where an improvement occur and with the size of the improvement (compared to the original approximative analysis). In figure 7(d) we see that the tight analysis is having a higher average improvement than the exact analysis for small task sets. The reason we can get a higher average improvement is that the exact analysis finds *more* tasks with improvements whereas those tasks have a *low* improvement, hence the average for the exact becomes lower than the average for the tightened analysis. While not showed in this paper, the graphs where we varied the number of transactions from 1 to 10, with load 90% and tasks/transaction=5 shows similar results; the tight analysis is almost as good as the exact analysis.

In figures 7(e) to 7(h) we compare the calculated response-times of the original approximative analysis (baseline) and our tighter approximative analysis (Tight)

[6] The confidence intervals has been calculated for mean values of normal distributed samples. However, we have not analysed the distribution of the samples and hence the confidence intervals should be taken as good indicators of the confidence obtained in the simulations rather than statistically correct confidence intervals.

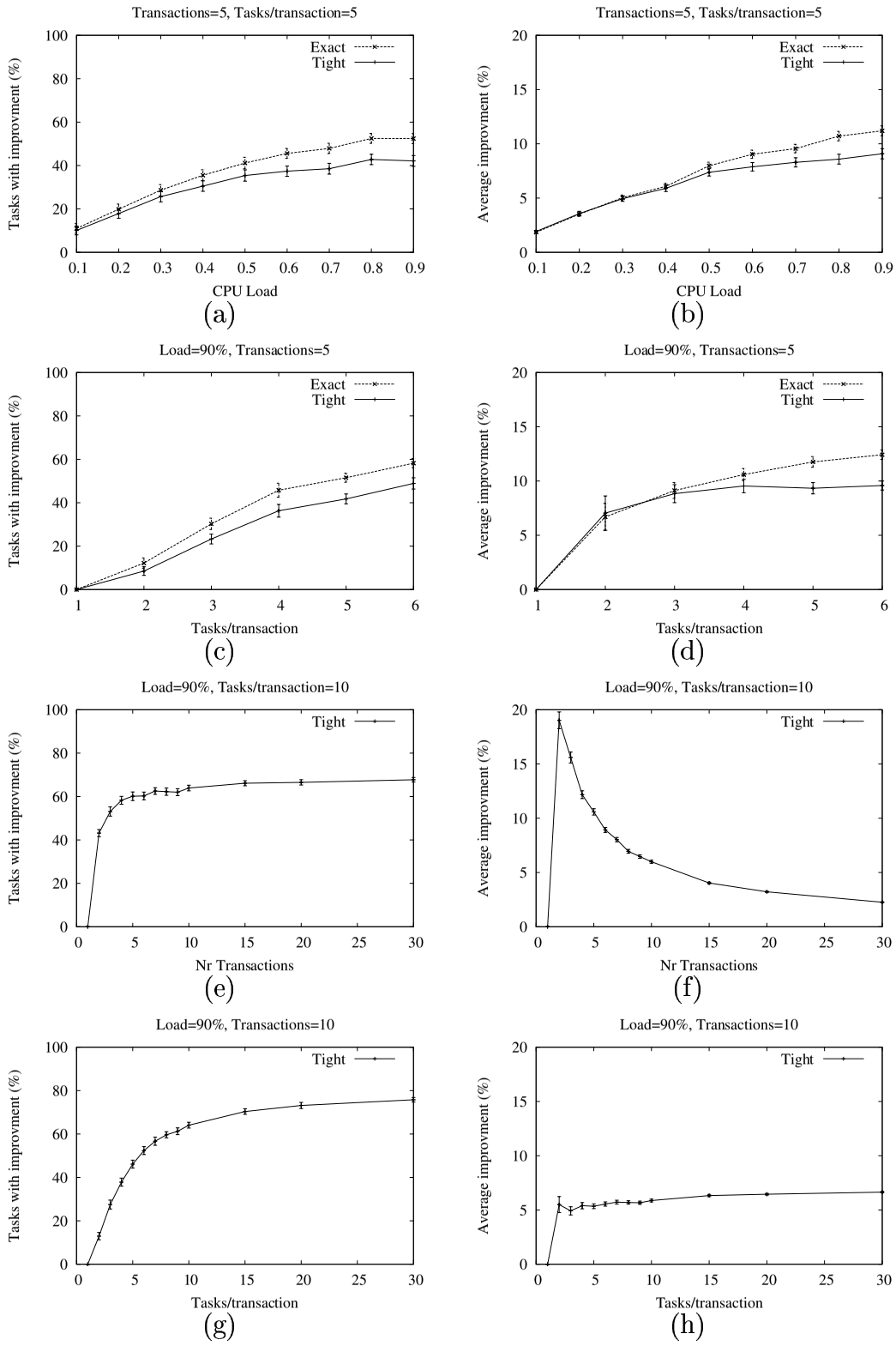


Figure 7: Simulation Results

for larger tasks sets. When the number of transaction grows, we see that the tighter analysis improves the response time for more than 70% of the tasks (figures 7(e) and 7(g)). However, as the number of transactions grow the average improvement drops to about 2% (figure 7(f)). While the graph is not shown (due to low statistical confidence), our simulations also show that the exact analysis approaches improvements of about 2%, which is also consistent with earlier evaluations of the approximative and the exact analysis [Tin94].

Varying the number of tasks/transaction we see that the average improvment is not really affected (figure 7(h)). This holds regardless of the number of transactions. Hence, if the number of transactions is set to 5 (note that figure 7(d) has 10 transactions) we get a 10% average improvement for more than 70% of the tasks.

# 6 Conclusions

We have presented an improvment to the existing approximative offset analysis for tasks with offsets, calculating significantly lower response times. This improvement comes from the fact that the original offset analysis considers an amount of interference, in interval $t$, constituting of the maximum amount of execution that is released for execution (stepped-stair interference) in $t$, whereas our method only accounts for the interference that can actually be experienced during $t$ (slanted- stair interference).

The original offset analysis can be compared to the round up problem of consecutive floating point calculations; where a round up in each calculation step yields an unneccesary high overestimation. Whereas, if the round up is postponed until after the last calculation step, the overestimation is minimised. This compares to our tighter method that does not use stepped-stair interference (rounded up values) before the maximisation function that is at the core of the approximative analysis.

Simulations show that our method follows the exact analysis quite close, i.e., our tighter analysis is almost as good as the exact analysis. Compared to the original approximative analysis there is a significante improvement, but the commonality and size of them are dependant on task set profile. In certain cases we can see an improvement up to 10% over the original analysis in up to 70% of the response time calculations.

# References


[ABD+95] N.C. Audsley, A. Burns, R.I. Davis, K. Tindell, and A.J. Wellings. Fixed Priority Pre-Emptive Scheduling: An Historical Perspective. *Real-Time Systems*, 8(2/3):129–154, 1995.

[ABT+93] N.C. Audsley, A. Burns, K. Tindell, M.F. Richardson, and A.J. Wellings. Applying New Scheduling Theory to Static Priority Preemptive Scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[But97] G.C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, 1997. ISBN 0-7923-9994-3.

[GH98] J. C. Palencia Gutierrez and M. Gonzalez Harbour. Schedulability Analysis for Tasks with Static and Dynamic Offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)*, December 1998.

[GH99] J. C. Palencia Gutierrez and M. Gonzalez Harbour. Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems. In *Proc. 20th IEEE Real-Time Systems Symposium (RTSS)*, pages 328–339, December 1999.

[JP86] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[LL73] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[MTS02] J. Mäki-Turja and M. Sjödin. Combining Dynamic and Static Scheduling in Hard Real-Time Systems. Technical Report MRTC no. 71, Mälardalen Real-Time Research Centre (MRTC), October 2002.

[Red03] O. Redell. Accounting for Precedence Constraints in the Analysis of Tree-Shaped Transactions in Distributed Real-Time Systems. Technical Report TRITA-MMK 2003:4, Dept. of Machine Design, KTH, 2003.

[SH98] M. Sjödin and H. Hansson. Improved Response-Time Calculations. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)*, December 1998. URL: http://www.docs.uu.se/~mic/papers.html.

[Tin92] K. Tindell. Using Offset Information to Analyse Static Priority Preemptively Scheduled Task Sets. Technical Report YCS-182, Dept. of Computer Science, University of York, England, 1992. Available at ftp://ftp.cs.york.ac.uk/pub/realtime/papers/YCS182_[12].ps.Z.

[Tin94] K. Tindell. *Fixed Priority Scheduling of Hard Real-Time Systems.* PhD thesis, University of York, February 1994. Available at ftp://ftp.cs.york.ac.uk/pub/realtime/papers/thesis/ken/.

# A Tighter Analysis for an Advanced Model

Here we present, using the model and method of Palencia Gutierrez *et al.* [GH98], our tighter analysis without the simplifying assumptions of section 2.2.

## A.1 The Advanced Model

First we briefly give Palencia Gutierrez *et al.*'s system model. Here a transaction $\Gamma_i$ is defined in the same was as in section 2.2:

$$\Gamma_i := \langle \{\tau_{i1}, \ldots, \tau_{in}\}, T_i \rangle$$

A task, $\tau_{ij}$, is defined by the following tuple:

$$\tau_{ij} := \langle C_{ij}, \phi_{ij}, D_{ij}, P_{ij}, B_{ij}, J_{ij} \rangle$$

Where $C_{ij}$ is the worst-case execution time, $\phi_{ij}$ is the offset, $D_{ij}$ the deadline, and $P_{ij}$ the priority. Also, $B_{ij}$ denotes the worst-case blocking of $\tau_{ij}$, i.e. the longest time that $\tau_{ij}$ can be delayed by a *lower* priority tasks (e.g. waiting for a semaphore to be released). Buttazzo gives several methods to calculate $B_{ij}$ [But97]. $J_{ij}$ denotes the maximum jitter of $\tau_{ij}$ (for a periodic task, the jitter is the maximum deviation from the periodicity, e.g., caused by a tick scheduler or obtained as a result of precedence constraints).

In addition to adding adding two new task-attributes, the following is also allowed:

- Offsets may be larger than period.
- Jitter may be larger than period.
- Deadline may be larger than period.

## A.2 Tighter Analysis

Our tightening is obtained by modifying a single equation in [GH98]. Equation 18 of Palencia Gutierrez *et al.* is as follows:

$$W_{ik}(\tau_{ab}, t) = \sum_{\forall j \in hp_i(\tau_{ab})} \left( \left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lceil \frac{t - \varphi_{ijk}}{T_i} \right\rceil \right) C_{ij} \qquad \text{(18 in [GH98])}$$

Where $W_{ik}(\tau_{ab}, t)$ denotes the interference $\Gamma_i$ poses on $\tau_{ab}$ during $t$ if $\tau_{ik}$ is the task in $\Gamma_i$ that coincides with the critical instant, i.e. it denotes for the extended

task model the same as $I(\tau_{ab}, \tau_{ik}, t)$ as explained on page 5. And $\varphi_{ijk}$ denotes the phasing between $\tau_{ij}$ and $\tau_{ik}$, i.e. it denotes the same as $phase(\tau_{ij}, \tau_{ik})$ as explained on page 5.

Equation 18 of Palencia Gutierrez *et al.* is modified in the same way as we modified equation 1 on page 5 to equation 5 on page 7. The result is:

$$W_{ik}(\tau_{ab}, t) = \sum_{\forall j \in hp_i(\tau_{ab})} \left( \left( \left\lfloor \frac{J_{ij} + \varphi_{ijk}}{T_i} \right\rfloor + \left\lfloor \frac{t - \varphi_{ijk}}{T_i} \right\rfloor + 1 \right) C_{ij} - x \right)$$
$$x = \begin{cases} 0 & t - \varphi_{ijk} < 0 \\ \max\left(0, C_{ij} - ((t - \varphi_{ijk}) \mod T_i)\right) & t - \varphi_{ijk} \geq 0 \end{cases} \tag{6}$$

## A.3 Discussion

The only thing we need to do to implement our tighter analysis is to smoothen out the stepped stairs caused by the original ceiling expression of Palencia Gutierrez *et al.*'s equation 18, and make that equation calculate slanted stairs, no further modification to their approach is needed. This is done with our modification in equation 6. Hence, all other equations of Palencia Gutierrez *et al.* [GH98] can be used unmodified.

Thus, the only extension in the advanced task model that really comes into play, as far as our tighter analysis is concerned, is the jitter ($J_{ij}$). The jitter is used explicitly in equation 6 but only contributes a constant value, hence it handled trivially. However, the jitter is also used implicitly in equation 6 through $\varphi_{ijk}$, which is defined in Palencia Gutierrez *et al.*'s equation 17 as:

$$\varphi_{ijk} = T_i - (\phi_{ik} + J_{ik} - \phi_{ij}) \mod T_i \qquad \text{(17 in [GH98])}$$

However, the exact definition of $\varphi_{ijk}$ (and hence the implicit use if the jitter) does not affect our tightening in equation 6, since $\varphi_{ijk}$ is used consistently in both the floor+1 expression and in the definition of $x$. Recall, the only purpose of $x$ is to smoothen out the stepped stairs, hence the only concern is that $x$ should be equal to 1 at the same $t$ as the floor+1 expression increases one step.

All other extensions to the system model of section 2.2 presented in section A.1 are handled by other equations presented by Palencia Gutierrez *et al.* [GH98]. That is, model features such as blocking and long deadlines and jitter does not affect the interference functions ($I(\tau_{xy}, \tau_{ic}, t)$ and $W_{ik}(\tau_{ab}, t)$) instead they affect the response time calculation (equation 4 on page 6), which is given as equation 28 by Palencia Gutierrez *et al.*.