Faster Response Time Analysis of Tasks With Offsets

Jukka Mäki-Turja Mikael Nolin
Mälardalen Real-Time Research Centre (MRTC)
Västerås, Sweden

E-mail: jukka.maki-turja@mdh.se

Abstract

We present a method that enables an efficient implementation of the approximative response time analysis for tasks with offsets presented by Tindell [7] and Palencia Gutierrez et al. [2].

The method utilises the fact that the interference a transaction imposes on a lower priority task exhibits a periodic and static pattern. This pattern can be pre-computed in a table, reducing the fix-point calculations to a simple lookup-up function. We show by simulations that the speed-up is substantial, and for realistically sized task sets, more than 600 times, compared to the original analysis.

1 Introduction

A powerful and well established schedulability analysis technique is the *Response-Time Analysis* (RTA) [1]. RTA is applicable to systems where tasks are scheduled in strict priority order which is the predominant scheduling technique used in real-time operating systems today.

Tindell proposed an extension to the RTA that take timing offsets between tasks into account [7]. He 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. Later Palencia Gutierrez *et al.* [2] formalised, generalised and improved Tindells work.

In this paper we present a method that enables an efficient implementation of the approximative offset analysis given by Tindell [7] and Palencia Gutierrez *et al.* [2]. The method significantly speeds up the calculation of response times as we will show by simulations.

Paper Outline: In section 2 we revisit and restate the original offset RTA presented by [7] and [2]. In section 3 we present our new method and the changes to the response time formula and show how to perform the precalculations needed. Section 4 presents evaluations of the method, and finally section 5 concludes the paper and outlines future work.

[1] In the context of scheduling analysis, *safe* implies that no underestimation is made.

2 Existing Offset RTA

2.1 System Model

The system model used is as follows: The system, Γ, consists of a number of transactions $\Gamma_1, \ldots, \Gamma_k$. A transaction, Γ_i, is defined by a set of tasks $\tau_{i1}, \ldots, \tau_{in}$ and a period T_i: 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, τ_{ij}, is defined by a worst case execution time (C_{ij}), an offset (O_{ij}), a deadline (D_{ij}), and a priority (P_{ij}). This is formally expressed as follows:

$$\begin{aligned}\Gamma :=& \{\Gamma_1, \ldots, \Gamma_k\} \\ \Gamma_i :=& \langle \{\tau_{i1}, \ldots, \tau_{in}\}, T_i \rangle \\ \tau_{ij} :=& \langle C_{ij}, O_{ij}, D_{ij}, P_{ij} \rangle\end{aligned}$$

The offset denotes the earliest release time of a task relative to the start of its 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.)

In this paper, the objective is to present our method for a simple task model making our contribution clear. Therefore, for presentation purposes, we introduce some simplifying assumptions. However, our method is directly applicable without most of these simplifications (with jitter as the one exception, see section 5 for a discussion on the matter). In this paper we assume:

- 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 are not modelled.
- Blocking on shared resources (e.g. using semaphores) is not modelled.
- Unique priority for each task is assumed, i.e. $P_{ij} \neq P_{kl}$.

2.2 Existing RTA Formula

For each task in Γ, RTA calculates an upper bound on the response time. We use τ_{xy} to denote the *task under analysis*, i.e., the task whos response time we are currently calculating. The major impact for a tasks response time is the interference caused by higher priority tasks. This interference is dependant on which task(s) coincide with the critical instant (CI). The traditional CI (for tasks without offsets) occurs when all higher priority tasks are released at the release of τ_{xy}.

To analyse tasks with offsets the notion of CI is modified to: at least one task out of every transaction is assumed to coincide with the critical instant [7]. For instance, consider the transaction depicted in figure 1. It consists of two tasks, τ_{i1} and τ_{i2}, and is defined as:

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

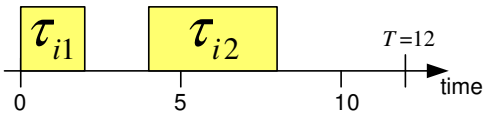

Figure 1. Example transaction with two tasks

In this example there are two candidate tasks to coincide with the critical instant. Figures 2(a) and 2(b) shows how interference, imposed on a lower priority tasks, grows over time for the two candidates. (Note that since the transaction is periodic, the depicted patterns repeat themselves after one period.)

The interference visualised in figures 2(a) and 2(b) is formally expressed as follows: For each candidate task, τ_{ic}, in transaction Γ_i, that could coincide with the critical instant, we calculate the amount of interference Γ_i imposes on τ_{xy} during a time interval of length t. We call that interference $I(\tau_{xy}, \tau_{ic}, t)$ and define it as:

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

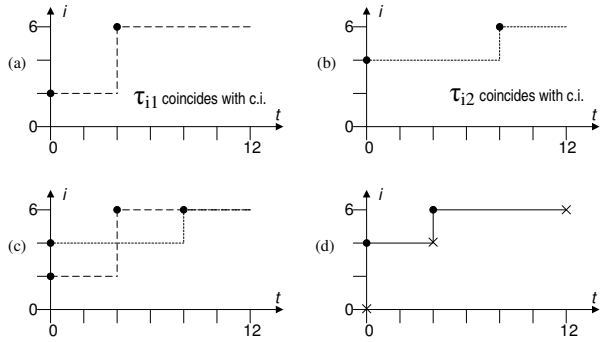

Figure 2. Interference caused by our example transaction

where $hp_i(\tau_{xy})$ represents the set of tasks belonging to transaction Γ_i with priority greater than the priority of τ_{xy}. $\mathit{phase}(\tau_{ic}, \tau_{ij})$ describes the distance in time from the release of τ_{ic} to the release of τ_{ij}, defined as:[2]

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

In equation 1 t' represents the time during which τ_{ij} has had a chance to interfere with τ_{xy}. Put in another way, t' "starts ticking" once t has reached beyond the offset O_{ij} (Bear in mind that τ_{ic} is released at $t = 0$).[3]

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 given by [7] and [3] defines one single, upward approximated, function for the interference caused by transaction Γ_i:

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

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:

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

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

3 Efficient Offset RTA

When calculating R_{xy}, the function $A(\tau_{xy}, \Gamma_i, t)$ (equation 3) will be evaluated repeatedly. For each task and

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

[3]Note that $\forall t : t' > -T_i$, hence the ceiling expression in equation 1 will always be greater than or equal to zero.

transaction pair (τ_{xy} and Γ_i) many different time-values, t, will be used during the fix-point iteration. However, since $A(\tau_{xy}, \Gamma_i, t)$ has a pattern that is repeated every T_i time unit, we could save a lot of computation if we could represent the interference function statically and during response-time calculation use a simple lookup function to obtain its value. We will in this section show how the response time formula changes using this pre-computed information and how to calculate and store that information.

3.1 Formula with Lookup Function

Again, consider our example transaction of figure 1. The interference, $A(\tau_{xy}, \Gamma_i, t)$, the transaction poses on a lower priority task, within $t = 0..T_i$, is depicted in figure 2(d).

To statically represent this function it is sufficient to store one point per "stair". In our case we store the concave corners, marked with crosses in figure 2(d), of $A(\tau_{xy}, \Gamma_i, t)$. In order to do that we define two arrays C_i^{pre} and t_i^{pre}. $C_i^{pre}[k]$ represents the maximum amount of interference Γ_i will pose on a lower priority task during interval lengths up to $t_i^{pre}[k]$.[4]

Using the two arrays we redefine the approximated interference function (equation 3 on the page before) as follows:

$$\begin{aligned} A(\tau_{xy}, \Gamma_i, t) =& \textit{full\_periods} * C_i^{pre}[|C_i^{pre}|] + C_i^{pre}[k] \\ \textit{full\_periods} =& t \text{ div } T_i \\ \textit{remaining\_t} =& t \text{ rem } T_i \\ k =& \min\{x \ : \ \textit{remaining\_t} \leq t_i^{pre}[k]\} \end{aligned} \tag{5}$$

The intuition behind equation 5 is that the interference during time interval t is calculated by two terms:

- Number of full periods (T_i) that fit within t, multiplied with the sum of all tasks C_{ij} in the transaction (stored in the last element of C_i^{pre}).
- Next, we look up the smallest time interval in t_i^{pre} that is longer than (or equal to) $remaining\_t$. The corresponding C_i^{pre} is the maximum interference during *remaining_t*.

Using equation 5 instead of equation 3 to compute R_{xy} (equation 4 on the preceding page) will significantly reduce the time to compute the response times as we will show in section 4.

[4]Note that C_i^{pre} and t_i^{pre} only should contain interference from *higher priority* tasks. Hence, they should be calculated with the subset of the tasks of Γ_i defined by $hp_i(\tau_{xy}) \cap \Gamma_i$. However, for simplicity, in this presentation we assume that all tasks of Γ_i are included in $hp_i(\tau_{xy})$.

3.2 Precomputing C_i^{pre} and t_i^{pre}

Lets see how the arrays C_i^{pre} and t_i^{pre} can be calculated. First, we have to consider each task τ_{ic} in Γ_i as a candidate to coincide with the critical instant.[4] For each candidate task τ_{ic} we define a set of points p_{ic}. Each point $p_{ic}[k]$ has an x and and y coordinate, and describes how the interference grows over time if τ_{ic} coincides with the critical instant. The poinst in p_{ic} corresponds to the convex corners of $I(\tau_{xy}, \tau_{ic}, t)$ (illustrated by dots in figures 2(a) and 2(b)). We define p_{ic} as:

$$\begin{aligned} p_{ic}[k].x =& O_{ik} - O_{ic} \mod T_i \\ p_{ic}[k].y =& \sum_{l=0}^{k} C_{ij}, \text{ where } j = (c+l) \mod |\Gamma_i| \\ & k \in 1 \ldots |\Gamma_i| \end{aligned}$$

For our example transaction of figure 1, we get the following two p_{ic}-s (corresponding to the dots in graphs 2(a) and 2(b) respectively):

$$\begin{aligned} p_{i1} &= [<0,2>, <4,6>] \quad \text{Figure 2(a)} \\ p_{i2} &= [<0,4>, <8,6>] \quad \text{Figure 2(b)} \end{aligned}$$

Now, we have all the information generated by the $I(\tau_{xy}, \tau_{ic}, t)$-functions, stored in the p_{ic}-sets. We define the set of points, p_i, as the union of all p_{ic}-s:

$$p_i = \bigcup_{\tau_{ic} \in \Gamma_i} p_{ic}$$

Graphically, the set p_i is illustrated by the dots in figure 2(c).

Next, will have to remove from p_i the points that will not be used to represent the approximation function $A(\tau_{xy}, \Gamma_i, t)$. This is done by the following algorithm:

```
sort p_i by increasing x-values
delete each p_ij where
    ∃k : p_ij.x == p_ik.x ∧ p_ij.y ≤ p_ik.y;
delete each p_ij where p_ij.y ≤ p_i(j−1).y;
```

Now, p_i contains the convex corners of the function $A(\tau_{xy}, \Gamma_i, t)$ (illustrated by the dots in figure 2(d)). All we have to do now is to find the concave corners (illustrated by the crosses in figure 2(d)) and store them in the arrays C_i^{pre} and t_i^{pre}. This is done by the followin algorithm:

```
C_i^pre[0] := t_i^pre[0] := 0
for k := 1 to |p_i| do
    C_i^pre[k] := p_i[k].y
    if k < |p_i| then
        t_i^pre[k] := p_i[k + 1].x
    else
        t_i^pre[k] := T_i
done
```

For our example transaction this gives the folloving C_i^{pre} and t_i^{pre} (corresponding to crosses in figure 2(d)):

$$\begin{aligned} C_i^{pre} &= [0,\ 4,\ 6] \\ t_i^{pre} &= [0,\ 4, 12] \end{aligned}$$

4 Evaluation

In order to evaluate the speed up, we have implemented both the original approximative response-time analysis (Old RTA) and our method including the pre-calculation step (Fast RTA). We have also implemented a task generator where we can vary system load, the number of transactions and the number of tasks. To fully describe the simulation setup and all results is beyond the scope of this WiP paper. The results are plotted together with the 95% confidence interval.

Figure 3 shows one simlation example, plotted both with a linear and a logarithmic y-axis. It shows that our proposal significantly speeds up the RTA calculations, especially when task sets grow beyond a trivial number of tasks and/or transactions. While difficult to judge from these graphs, the speedup factor is over 600 times (for 50 tasks/transaction, 10 transactions and 90% load), where the old analysis takes about 21 seconds and our fast RTA takes about 30 milliseconds. Also, reducing execution times from the 10s of seconds range to the 1/100th of a second range has significant practical benefit (e.g. it enables the use of our method inside the inner loop of an optimisation algorithm, such a priority assignment algorithm).

The graphs presented are consistent with our other simulations when varying both the number of task and the number of transactions. However, varying the load gave, as expected, no significant improvement for our method.

5 Conclusions and Future Work

We have presented a novel method for implementing the approximative response time analysis for tasks with offsets presented by Tindell [7] and Palencia Gutierrez *et al.* [2]. The novelty in this approach is that the interference a transaction imposes on lower priority tasks, is pre-calculated and statically stored in two arrays. The advantage of this approach is that it substantially reduces the time to calculate response times as we have shown in simulations. The enabling factor that makes it possible to pre-calculate this information is that the interference exhibits a regular periodic pattern.

Extending our task model to that of [2] is straightforward (using the same techniques and equations as [2]). The one exception is the inclusion of jitter in the model. Accounting for jitter is somewhat more problematic since the interference no longer exhibit a single repetitive pattern. In the full version of this paper we will address three additional issues:

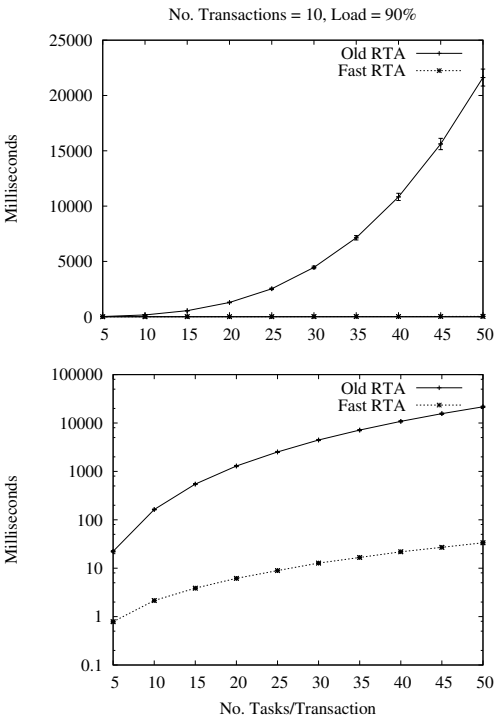

Figure 3. Execution times for RTA

(1) adding jitter to our model, (2) show how this speed-up can be applied to our tighter RTA (that calculates lower response times) [4, 5], and (3) extend our evaluations.

References

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

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

[3] J. C. P. Gutierrez and M. G. 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.

[4] J. Mäki-Turja and M. Sjödin. Improved Analysis for Real-Time Tasks With Offsets. Submitted for publication, August 2003.

[5] J. Mäki-Turja and M. Sjödin. Improved Analysis for Real-Time Tasks With Offsets – Advanced Model. Technical Report MRTC no. 101, Mälardalen Real-Time Research Centre (MRTC), May 2003.

[6] 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.

[7] K. Tindell. Using Offset Information to Analyse Static Priority Pre-emptively 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.