On Sustainability for Offset Based Response-Time Analysis

Jukka Mäki-Turja
Mälardalen University
Västerås, SWEDEN
jukka.maki-turja@mdh.se

Kaj Hänninen
Mälardalen University
Västerås, SWEDEN
kaj.hanninen@mdh.se

Mikael Sjödin
Mälardalen University
Västerås, SWEDEN
mikael.sjodin@mdh.se

ABSTRACT

It is known that offsets in general are not sustainable, i.e., if offsets in a system are modified it is not trivial to determine if this affects schedulability or not. In this paper we investigate if there are situations where offsets can be deemed sustainable, i.e., situations where one can safely modify an offset without impairing schedulability.

We introduce, for the approximate Response-Time Analysis, a concept of a subsumes relation between demand bound functions and show that if offsets are modified in such a way that the modified demand bound function is subsumed by the original demand bound function, then the modification is sustainable.

We show an example of how this information can be used in an engineering context where all system parameters are fixed but offsets. We generate all transaction that are subsumed by a schedulable transaction and merge this information in a presentable form for the engineer. This information can be used to safely modify the transaction or to investigate the robustness of the schedulability.

KEYWORDS

scheduling sustainability, real-time, response-time analysis

ACM Reference Format:
Jukka Mäki-Turja, Kaj Hänninen, and Mikael Sjödin. 2021. On Sustainability for Offset Based Response-Time Analysis. In *7th Conference on the Engineering of Computer Based Systems (ECBS 2021), May 26–27, 2021, Novi Sad, Serbia.* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

The notion of schedulability is well understood within the real time systems community. A system is specified according to some formal notation, and is deemed schedulable with respect to a scheduling policy if it is guaranteed to meet all its timing requirements (typically expressed as deadlines). So schedulability implies that all deadlines are satisfied if the system behaves according to its parameterized specification.

Response-Time Analysis (RTA) [1, 11] is a powerful and well established schedulability analysis technique. RTA is a method to calculate upper bounds on response-times for tasks in real-time systems. In essence RTA is used to perform a schedulability test, i.e., checking whether or not tasks in the system will satisfy their deadlines.

ECBS 2021, May 26–27, 2021, Novi Sad, Serbia

ACM ISBN 978-1-4503-9057-6/21/05…$15.00

To be able to calculate less pessimistic response times in systems where tasks may have dependencies in their release times, Tindell introduced RTA for a task model with offsets, the transactional task model, [12]. Palencia Gutiérrez and González Harbour formalized and extended the work of Tindell in [10]. And in [9] Mäki-Turja and Nolin introduced a *fast-and-tight* RTA method based on their work.

In [3, 4] Burns and Baruah introduced the concept of a schedulability test being sustainable. A given scheduling test is sustainable if any system that is schedulable under its worst-case specification remains so when its behavior is "better" than worst-case. Meaning that if the system behaves better than its parameterized specification it will also be schedulable.

They also gave some parameters, such as *Worst Case Execution Time*, that typically are sustainable for most schedulability tests. However, the offset parameter used in [9, 10, 12] is not generally sustainable. That is, there is no way of knowing how to change offset parameters "to the better" so that system remain schedulable. Quoting their paper:

> "That is, schedulability tests that do not "ignore" these parameters are generally not sustainable in that task systems deemed schedulable cease to be so if not just these parameters change, but also if other parameters such as the deadline or period change "for the better". It seems that the safe way to analyse systems with offsets and best-case execution times is to ignore them (i.e., assume that all these parameters are equal to zero)."

Since then, the sustainability analysis has been studied in e.g. [2, 5].

In this paper we will show that the approximate RTA for task with offset presented in [10, 12] and especially the *fast-and-tight* RTA method [9] can be a base for finding sustainability conditions for offsets.

2 TASKS WITH OFFSETS RTA

This section revisits the response-time analysis for tasks with offsets [9, 10, 12] and illustrates some intuition behind the analysis and the formulae.

2.1 System model

The system model used is as follows: The system, Γ, consists of a set of k transactions $\Gamma_1, \ldots, \Gamma_k$. Each transaction Γ_i is activated by a periodic sequence of events with period T_i (for non-periodic events T_i denotes the minimum inter-arrival time between two consecutive events). The activating events are mutually independent, i.e., phasing between them is arbitrary. A transaction, Γ_i, contains $|\Gamma_i|$

tasks, and each task may not be activated (released for execution) until a time, *offset*, elapses after the arrival of the external event.

We use τ_{ij} to denote a task. The first subscript denotes which transaction the task belongs to, and the second subscript denotes the number of the task within the transaction. A task, τ_{ij}, is defined by a worst case execution time (C_{ij}), an offset (O_{ij}), a deadline (D_{ij}), maximum jitter (J_{ij}), maximum blocking from lower priority tasks (B_{ij}), and a priority (P_{ij}). The system model is formally expressed as follows:

$$\Gamma := \{\Gamma_1, \ldots, \Gamma_k\}$$
$$\Gamma_i := \langle \{\tau_{i1}, \ldots, \tau_{i|\Gamma_i|}\}, T_i \rangle$$
$$\tau_{ij} := \langle C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij} \rangle$$

There are no restrictions placed on offset, deadline or jitter, i.e., they can each be either smaller or greater than the period.

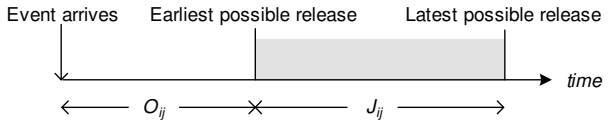

Figure 1: Relation between an event arrival, offset, jitter and task release

The relation between event arrival, offset, jitter and task release is graphically visualized in Fig. 1. After the event arrival, task τ_{ij} is not released for execution until its offset (O_{ij}) has elapsed. The task release may be further delayed by jitter (maximally until $O_{ij} + J_{ij}$) making its exact release uncertain. For a more extensive explanation of task parameters see [10]. Parameters for an example transaction (Γ_i) with two tasks (τ_{i1}, τ_{i2}) are depicted in Fig. 2.

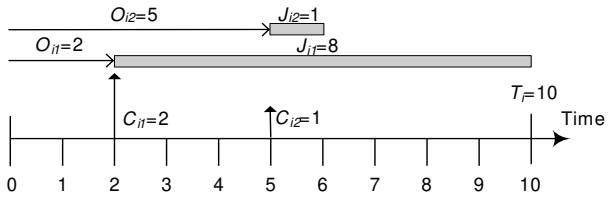

Figure 2: An example transaction Γ_i

2.2 Response-time analysis

The goal of RTA is to facilitate a schedulability test for each task in the system by calculating an upper bound on its worst case response-time. We use τ_{ua} (task a, belonging to transaction Γ_u) to denote the *task under analysis*, i.e., the task which response time we are currently calculating.

In the classical RTA (without offsets) the *critical instant* for τ_{ua} occurs when it is released at the same time as all higher priority tasks [6, 7]. In a task model with offsets this assumption yields pessimistic response-times since some tasks cannot be released simultaneously due to offset relations. Therefore, Tindell [12] relaxed the notion of critical instant to be:

> At least one task in every transaction is to be released at the critical instant. (Only tasks with priority higher or equal to τ_{ua} are considered.)

Since it is not known which task coincides with (is released at) the critical instant, every task in a transaction must be treated as a *candidate* to coincide with the critical instant.

Tindell's exact RTA tries every possible combination of candidates among all transactions in the system. This, however, becomes computationally intractable for anything but small transactions (the number of possible combinations of candidates is m^n for a system with n transactions and with m tasks per transaction). Therefore Tindell provided an approximate RTA that still gives good precision but uses one single approximation function for each transaction. Palencia Gutiérrez and González Harbour [10] formalized and generalized Tindell's work. And in [9] Mäki-Turja and Nolin further extended the work of Palencia Gutiérrez and González Harbour by introducing a *fast-and-tight* RTA method. We will be using the formalism presented in [9].

2.3 Interference function

Central to RTA is to capture the worst case interference a higher or equal priority task (τ_{ij}) causes the task under analysis (τ_{ua}) during an interval of time t. Since a task can interfere with τ_{ua} multiple times during t, we have to consider interference from possibly several *instances*. The interfering instances of τ_{ij} can be classified into two sets:

*Set*1 Activations that occur before or at the critical instant and that can be delayed by jitter so that they coincide with the critical instant.

*Set*2 Activations that occur after the critical instant

When studying the interference from an entire transaction Γ_i, we will consider each task, $\tau_{ic} \in \Gamma_i$, as a *candidate* for coinciding with the critical instant.

RTA for tasks with offsets is based on two fundamental theorems:

(1) The worst case interference a task τ_{ij} causes to τ_{ua} is when *Set*1 activations are delayed by an amount of jitter such that they all occur at the critical instant and the activations in *Set*2 have zero jitter.
(2) The task of Γ_i that coincides with the critical instant (denoted τ_{ic}), will do so after experiencing its worst case jitter delay.

In order to determine the amount of *Set*2 interference for a task, τ_{ij}, we need to know when the first activation of τ_{ij} occurs after the critical instant. This phasing between a task, τ_{ij}, and the critical instant, which according to theorem 2 occurs at $O_{ic} + J_{ic}$, becomes:

$$\Phi_{ijc} = (O_{ij} - (O_{ic} + J_{ic})) \bmod T_i \quad (1)$$

Fig. 3 illustrates the four (two transactions and two critical instant candidates) different Φ_{ijc}-s that are possible for our example transaction in Fig. 2. Note that the time of origin is set at the critical instant. The upward arrows denote task releases. The height of the upward arrows denotes the amount of execution released.
Fig. 3(a) shows, for the case when τ_{i1} coincides with the critical instant, the invocations in *Set*1 (arriving at time 0) and the first invocation in *Set*2. Fig. 3(b) shows the corresponding situation when τ_{i2} is the candidate to coincide with the critical instant.

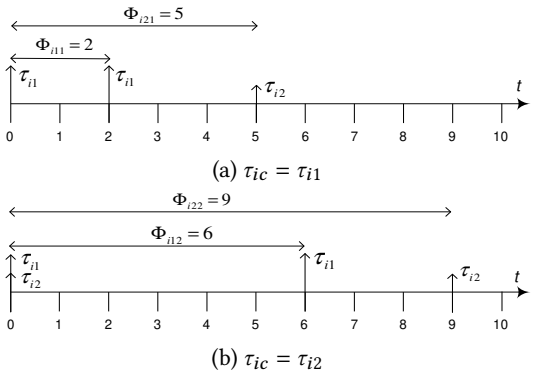

Figure 3: Φ-s for the two candidates in Γ_i

Given the two sets of task instances ($Set1$ and $Set2$) and the corresponding phase relative to the critical instant (Φ_{ijc}), the interference caused by task τ_{ij} can be divided into two parts:

(1) The part caused by instances in $Set1$ (which is independent of the time interval t), I_{ijc}^{Set1}, and
(2) the part caused by instances in $Set2$ (which is a function of the time interval t), $I_{ijc}^{Set2}(t)$.

These are defined as follows:

$$I_{ijc}^{Set1} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor C_{ij}$$
$$I_{ijc}^{Set2}(t) = \left\lceil \frac{t^*}{T_i} \right\rceil C_{ij} - x$$
$$t^* = t - \Phi_{ijc} \tag{2}$$
$$x = \begin{cases} 0 & t^* \leq 0 \\ 0 & t^* \bmod T_i = 0 \\ 0 & t^* \bmod T_i \geq C_{ij} \\ C_{ij} - (t^* \bmod T_i) & \text{otherwise} \end{cases}$$

The worst case interference transaction Γ_i poses on τ_{ua}, during a time interval t, when candidate τ_{ic} coincides with the critical instant, is:

$$W_{ic}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(I_{ijc}^{Set1} + I_{ijc}^{Set2}(t) \right) \tag{3}$$

Where $hp_i(\tau_{ua})$ denotes tasks belonging to transaction Γ_i, with priority higher or equal to the priority of τ_{ua}.

2.4 Approximation function

Since we beforehand cannot know which task in each transaction coincides with the critical instant, the exact analysis tries every possible combination [10, 12]. However, since this is computationally intractable for anything but small task sets, the approximate analysis defines one single, upward approximated, function for the interference caused by transaction Γ_i [10, 12]:

$$W_i^*(\tau_{ua}, t) = \max_{\forall c \in hp_i(\tau_{ua})} W_{ic}(\tau_{ua}, t) \tag{4}$$

That is, $W_i^*(\tau_{ua}, t)$ simply takes the maximum of each interference function (for each candidate τ_{ic}).

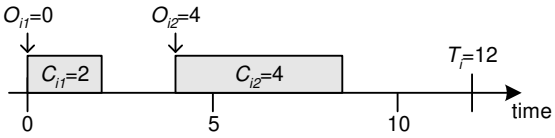

Figure 4: A simple example transaction

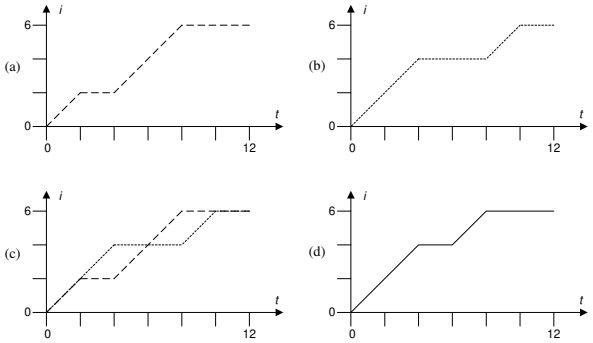

Figure 5: Interference *imposed* on τ_{ua} by our example transaction

As an example, consider transaction Γ_i depicted in Fig. 4 (Jitter is set to 0). Fig. 5 shows the interference functions (W_{i1} and W_{i2}) for the two candidates, and it shows how W_i^* is derived from them by taking the maximum of the two functions at every t. Given the interference (W_i^*) each transaction causes, during a time interval of length t, the response time of τ_{ua} (R_{ua}) can be calculated. The complete set of RTA formulae, and how W_i^* is used, can be found in in [9].

3 SUSTAINABILITY IN RTA WITH OFFSETS

Before defining the criterion for sustainability in RTA with offsets we will recapitulate the "subsumes" relation introduced in [9], since this relation is also used to define if a transaction Γ_a is sustainable w.r.t. another transaction Γ_b.

3.1 Subsumed points and calculation of approximation function

When calculating response times, the function $W_i^*(\tau_{ua}, t)$ will be evaluated repeatedly. However, since $W_i^*(\tau_{ua}, t)$ has a repetitive pattern, a lot of computational effort could be saved by representing the interference function statically, and during response time calculation use a simple lookup function to obtain its value. In [9] they showed how to create and use such a lookup function.

[9] defined a set of points, p_i, where each point $p_i[a]$ has an x (representing time) and a y (representing interference) coordinate, describing how the interference from transaction i grows over time. The points in p_i correspond to the convex corners of $W_i^*(\tau_{ua}, t)$ of eq. 4.

In order to determine the points in p_i corresponding to the convex corners of $W_i^*(\tau_{ua}, t)$, [9] defined a *subsumes* relation: A point

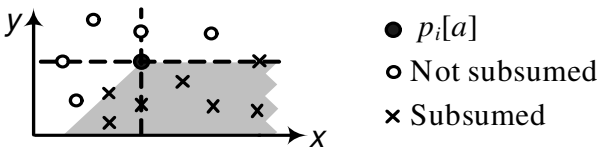

Figure 6: The subsumes relation

$p_i[a]$ subsumes a point $p_i[b]$ (denoted $p_i[a] \succ p_i[b]$) if the presence of $p_i[a]$ implies that $p_i[b]$ is not a convex corner. Fig. 6 illustrates this relation graphically for a point $p_i[a]$ with a shaded region, and the formal definition is:

$$p_i[a] \succ p_i[b] \text{ iff}$$
$$p_i[a].y \geq p_i[b].y \wedge \left(p_i[a].x - p_i[a].y \leq p_i[b].x - p_i[b].y\right)$$

In [9] they showed that storing the convex corners of $W_i^*(\tau_{ua}, t)$ for one period (or two periods if tasks have jitter) suffices to represent $W_i^*(\tau_{ua}, t)$.

3.2 Subsumed demand bound functions

In this paper we define a subsumes relation for a *demand bound function* (dbf) represented by its set of convex corners. Since $W_i^*(\tau_{ua}, t)$ is a demand bound function for transaction Γ_i which can be represented by its convex corners we define:

$$convex(dbf) \equiv W_i^*(\tau_{ua}, t) \tag{5}$$

Following the reasoning of subsumed points ($p_a \succ p_b$) in the previous section, a demand bound function dbf_a subsumes another demand bound function dbf_b if and only if all convex corners of dbf_b are subsumed by some convex corner of dbf_a. The formal definition is as follows:

$$dbf_a \succ dbf_b \text{ iff}$$
$$\forall p_b \in convex(dbf_b) \exists p_a \in convex(dbf_a) : p_a \succ p_b$$

Intuitively, this definition means that in a subsumed dbf_b all its points are in the shaded area of some point in bdf_a in Fig. 6. This also means that dbf_b is lower (or equal) than dbf_a for times t.

3.3 Sustainable replacement or modification of a transaction

Assuming that a task τ_{ua} is schedulable with $W_i^*(\tau_{ua}, t)$. Then replacing Γ_i (and its $W_i^*(\tau_{ua}, t)$) with any Γ_j (with $W_j^*(\tau_{ua}, t)$) still renders τ_{ua} schedulable if the new $W_j^*(\tau_{ua}, t)$ is not greater than $W_i^*(\tau_{ua}, t)$ for any time t.

This is the equivalent to testing if the demand bound function of Γ_j is subsumed by that of Γ_i. Formally:

> A transaction Γ_i can be replaced by another transaction Γ_j without rendering τ_{ua} unschedulable if $dbf_i \succ dbf_j$.

Informally, this means that we can modify parameters of a transaction Γ_i and create any transaction Γ_j without affecting schedulability adversely if $dbf_i \succ dbf_j$. In this case we say that Γ_j is sustainable w.r.t. Γ_i.

Table 1

Transaction			
Task	C	O	J
τ_1	3	0	0
τ_2	2	5	0
τ_3	1	10	0
T = 15			

4 EXAMPLE SECTION

In this section we will show how to use the subsumes relation for demand bound functions in order to determine how offsets can change and the transaction still be deemed sustainable.

We have investigated the impact of jitter. However, in this paper we will assume zero jitter for tasks in a transaction. The main result of this paper is still valid under non-zero jitter assumption. Non-zero jitter, results in a computational complexity that still needs to be further investigated

Most common application for the transactional task model (tasks with offsets) is the one for distributed systems and holistic scheduling. However, another use is to use it in a hybrid, static and dynamic, scheduling context as in [8]. Thus the zero jitter assumption is valid in such an engineering context. We will also assume integer values for task parameters.

We will, as an example use the transaction of table 1.

The three resulting $W_{ic}(\tau_{ua}, t)$ can be seen in Fig. 7, where $W_i^*(\tau_{ua}, t)$ is obtained by taking the maximum at each t.

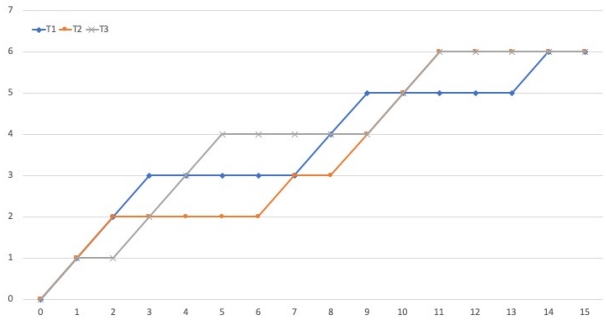

Figure 7: The three $W_{ic}(\tau_{ua}, t)$ of our example transaction

In order to determine all possible offsets (which is a generic combinatorial problem) that result in a sustainable transaction we generate all *unique* transactions where we keep all system parameters fixed besides offsets. With unique we mean that a transaction just shifting offsets, while retaining their relative distance between the tasks in the transaction, will not be considered unique. Just shifting offsets, while keeping their relative distance, will not change the resulting $W_i^*(\tau_{ua}, t)$. As an example take offsets $\tau_1.o = 1$, $\tau_2.o = 6$, $\tau_3.o = 11$, the resulting individual $W_{ic}(\tau_{ua}, t)$ and thus $W_i^*(\tau_{ua}, t)$ will not change because the relative order between tasks is kept.

For every new transaction we apply the subsumes relation in order to determine if it is sustainable w.r.t. the original transaction. We generate every possible offset combination such that:

$$\left(\langle\{\tau_1, \tau_2, \ldots \tau_n\}, T\rangle \mid \tau_1.o, \tau_2.o, \ldots \tau_n.o \in [0..T]\right)$$

If precedence relations should be kept among the tasks the possible offset combinations reduces to:

$$\big(\langle\{\tau_1, \tau_2, \ldots \tau_n\}, T\rangle \mid \tau_1.o, \tau_2.o, \ldots \tau_n.o \in [0..T] \wedge (\tau_1.o \leqslant \tau_2.o \leqslant \ldots \leqslant \tau_n.o)\big)$$

The following table shows the result of our example transaction (only the new offsets of the tasks are shown):

Table 2

Resulting unique transactions			
#	O_1	O_2	O_3
1	0	5	10
2	0	6	10
3	0	6	11
4	0	6	12
5	0	7	10
6	0	7	11
7	0	7	12
8	0	9	5
9	0	9	6
10	0	9	7
11	0	10	5
12	0	10	6
13	0	10	7
14	0	11	6
15	0	11	7
16	4	9	0

Table 2 shows all the possible unique variations of offsets that are sustainable w.r.t. the original transaction.

Take for instance the offset combination $\tau_1.o = 0$, $\tau_2.o = 7$, $\tau_3.o = 12$. Fig. 8 shows the three $W_{ic}(\tau_{ua}, t)$, whereas Fig. 9 shows that the resulting $W_i^*(\tau_{ua}, t)$ overlaid with our original $W_i^*(\tau_{ua}, t)$. One can see that the original $W_i^*(\tau_{ua}, t)$ subsumes the new $W_i^*(\tau_{ua}, t)$.

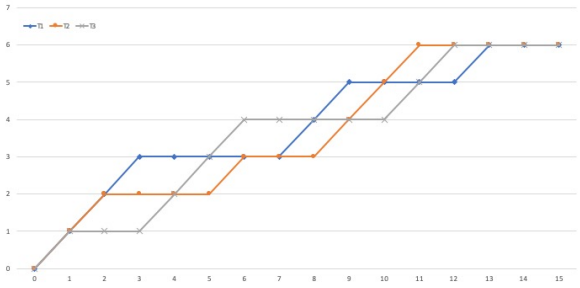

Figure 8: The three $W_{ic}(\tau_{ua}, t)$ of $\tau_1.o = 0$, $\tau_2.o = 7$, $\tau_3.o = 12$

We finally look at an example combination of $\tau_1.o = 0$, $\tau_2.o = 7$, $\tau_3.o = 13$. Its resulting $W_i^*(\tau_{ua}, t)$ can be seen not to be subsumed by our original $W_i^*(\tau_{ua}, t)$ in Fig. 10

In order to have a better overview on how offsets can be changed we merge the resulting transactions. This will result in the following information:

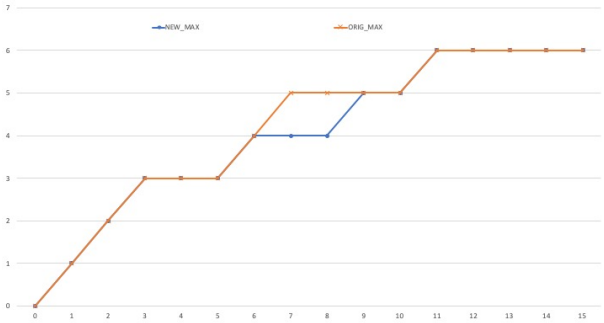

Figure 9: Subsumes relation with orginal $W_i^*(\tau_{ua}, t)$

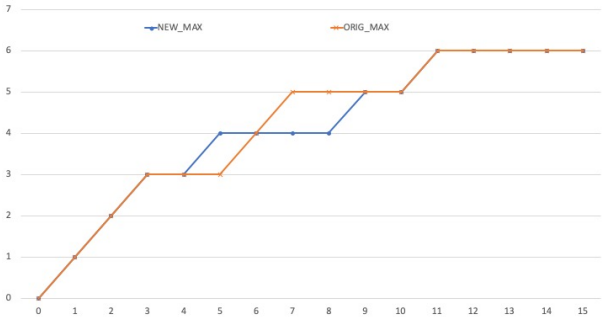

Figure 10: Subsumes relation with orginal $W_i^*(\tau_{ua}, t)$

Table 3

Merged unique transactions			
#	O_1	O_2	O_3
1	0	5	10
2-7	0	6-7	10-12
8-13	0	9-10	5-7
14-15	0	11	6-7
16	4	9	0

The interpretation of table 3 is as follows: τ_3 can have an offset of 10 as long as τ_1 has an offset of 0 and τ_2 has an offset of 5. τ_3 can have an offset between 10 and 12 as long as τ_1 has an offset of 0 and τ_2 has an offset that lies between 6 and 7. As an example one can shift the row one of the merged transaction by two resulting in offsets 2, 7, and 12 respectively and that is also a transaction.

One should note that the order one merge the transactions will result in different information. More on that in the discussion section.

5 DISCUSSION

In this section we will discuss the engineering context of the sustainability analysis and some interpretation details of our example transaction of the previous section.

The main use of the analysis presented in this paper is that it can be used as a sensitivity analysis at design time. That is, the engineer will be able to see how sensitive the system is to changes, in system parameters, at run-time and still be schedulable. Another use is, in

the hybrid scheduling context, to elaborate and make changes in a static schedule without doing a full rescheduling of the system.

In the example section we only considered unique transactions. We do it to make a more compact representation of information to the user.

Consider a sustainable transaction. Then, by just shifting offsets and keeping their relative distance will, trivially, result in a valid and sustainable transaction since the shifting will not change $W_i^*(\tau_{ua}, t)$. Therefore, the resulting transactions, as well as, the offsets of the merged transactions can be shifted, as long as their relative distance is retained.

In our example we chose to generate and merge the tasks in the following order, τ_3, τ_2, τ_1. The resulting table of merged transactions will look different depending of the order on generation and merging. However, since the task offsets can be shifted, the same information will be in that table, regardless what order one generates and merges transaction. As an example consider the following order instead, τ_2, τ_3, τ_1, then we get the merged transactions of table 4.

Table 4

Merged unique transactions			
#	O_1	O_2	O_3
1	0	5	10
2	0	9-10	5
3	0	9-11	6-7
4	0	5-7	11
5	0	6-7	12
6	8	0	3
7	9	0	4

Note that the information in table 3 and table 4 look very different. However, they represent the same sustainable transactions where one has just shifted offsets. For example row 2-7 in table 3 is represented by rows 4-7 in table 4 in the following manner: The transaction with offsets 0, 6, 10 in table 3 is represented by row 7, in table 4, where all tasks are shifted by T-9 time units. The transaction with offsets 0, 6, 11 in table 3 is represented by row 4, in table 4. The transaction with offsets 0, 6, 12 in table 3 is represented by row 5, in table 4. The transaction with offsets 0, 7, 10 in table 3 is represented by row 6, in table 4, where all tasks are shifted by T-8 time units. The transaction with offsets 0, 7, 11 in table 3 is represented by row 4. Transaction with offsets 0, 7, 12 in table 3 is represented by row 5.

From an engineering point of view one can focus on one task and represent the circumstances how its offset can change.

However, one can use the order as wanting to visualize the most important task first. In our example we chose τ_3 as the one that was most important and thus it exhibits most variability in the resulting merged transactions.

As can be seen we have placed no restrictions how to change offsets. In a distributed system context, for example, one would normally have precedence relations between the tasks. This will just yield in fewer (or equally many) valid transactions since one has to prune away those offsets that breaks the precedence relation. Thus, introducing precedence relations will result in a more restrictive analysis.

We chose to have zero jitter in our example just to highlight the effect of changing offsets. However, the analysis presented in this paper does not require zero jitter. In fact, the subsumes demand-bound function relation can be used for changing any system parameters. However, having non-zero jitter means that one must use the subsumes relation of the demand-bound function over two periods. The reason for this is that $W_i^*(\tau_{ua}, t)$ differs in first and subsequent periods. See [9] for further explanations.

6 CONCLUSIONS

Offsets in general are not sustainable, i.e., if offsets, or other system parameters, deviate from specification there is no way of guaranteeing schedulability for a given schedulability analysis. In this paper we investigate under what conditions offsets can be sustainable.

We show that the approximate RTA, for task with offsets, can be a base for finding sustainability conditions for offsets. We define a subsumes relation for a *demand bound function* (dbf) in order to define sustainability conditions for changed system parameters. Assuming that a transaction is schedulable with dbf_a, then any dbf_b (with changed system parameters) is also schedulable if the latter is subsumed by the former.

We exemplify how the subsumes relation can be used in an engineering context to perform sustainability analysis of a system where all system parameters, except offsets, are constant.

The main use of the sustainability analysis is that it can be used as a sensitivity analysis at design time. Another use is, in the hybrid scheduling context, in the sense that it can be used to elaborate and make changes in a static schedule without doing a full rescheduling of the system.

The proposed sustainability analysis, presented in this paper, is now being implemented in a commercial tool suite. The tool suite is being used to develop state of the art control systems such as heavy construction equipment. The intention of the tool is to support developers, at design time, by generating new $W_i^*(\tau_{ua}, t)$ when system parameters change. With such changes in $W_i^*(\tau_{ua}, t)$ the tool can perform an automatic sustainability control.

ACKNOWLEDGMENTS

This work was supported by the Swedish Research Council through the FAST-ARTS project.

REFERENCES

[1] N.C. Audsley, A. Burns, R.I. Davis, K. Tindell, and A.J. Wellings. 1995. Fixed Priority Pre-Emptive Scheduling: An Historical Perspective. *Real-Time Systems* 8, 2/3 (1995), 173–198.
[2] T. P. Baker and S. K. Baruah. 2009. Sustainable Multiprocessor Scheduling of Sporadic Task Systems. In *21st Euromicro Conference on Real-Time Systems*.
[3] S. Baruah and A. Burns. 2006. Sustainable Scheduling Analysis. In *27th IEEE International Real-Time Systems Symposium*.
[4] A. Burns and S. Baruah. 2008. Sustainability in Real-time Scheduling. *Journal of Computing Science and Engineering* 2, 1 (2008), 74–97.
[5] Z. Guo, S. Sruti, B. C. Ward, and S. Baruah. 2017. Sustainability in Mixed-Criticality Scheduling. In *IEEE Real-Time Systems Symposium*.
[6] M. Joseph and P. Pandya. 1986. Finding Response Times in a Real-Time System. *Comput. J.* 29, 5 (1986), 390–395.
[7] C. Liu and J. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM* 20, 1 (1973), 46–61.

[8] J. Mäki-Turja, K. Hänninen, and M. Nolin. 2005. Efficient Development of Real-Time Systems Using Hybrid Scheduling. In *International conference on Embedded Systems and Applications (ESA).*

[9] J. Mäki-Turja and M. Nolin. 2005. Fast and Tight Response-Times for Tasks with Offsets. In *Proc. of the 17th Euromicro Conference on Real-Time Systems.*

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

[11] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok. 2004. Real time scheduling theory: A historical perspective. *Real-Time Systems* (December 2004).

[12] K. Tindell. 1992. *Using Offset Information to Analyse Static Priority Pre-emptively Scheduled Task Sets.* Technical Report YCS-182. Dept. of Computer Science, University of York, England.