Using DIT to simplify the DBF feasibility test

Thomas Chapeaux

Supervisor
Laurent George, ECE Paris

Thanks to
Paul Rodriguez

July 2013

Contents

1 Real-Time Systems

$\tau = \{\tau_1, \cdots, \tau_n\}$
$\tau_i = (O_i, C_i, D_i, T_i)$
$H = LCM(T_i)$

2 The demand-bound function feasibility test

2.1 Definition of the demand-bound function

Definition 1. *The **demand-bound function (DBF)** ([1] and [2]), defined for a task set τ and noted $\mathrm{dbf}(t)$, is equal to the maximal cumulated execution time of jobs of τ contained in any interval of length t.*

Mathematically,

$$\mathrm{dbf}(t) = \max_{\{t_1, t_2 \,:\, t_2 - t_1 = t\}} \sum_{i=1}^{n} C_i \; h_i(t_1, t_2)$$

where $h_i(t_1, t_2)$ is the number of jobs of task i whose arrival and deadline are in (non-strict) the interval $[t_1, t_2]$.

As the synchronous arrival pattern is known to be the worst case, in that particular case we can only consider the interval $[0, t]$. In an asynchronous arrival pattern, it is sufficient to check only intervals where $t_1 < O_{max} + H$, as the values will be periodic after it.

In the sequel a closed form expression of the DBF is given in the synchronous case. Intuitively, in that case, it will be a step-function whose increase will occur at every job deadline, with the increase being equal to the execution time of the corresponding job.

Starting from the definition, it remains to find a method to compute the values of $h_i(t)$. In the synchronous case, these values can be obtained as

$$h_i(t) = \max \{k \in \mathbb{N} : kT_i + D_i \leq t\} + 1$$

Indeed, $k = 0$ correspond to the first task, $k = 1$ to the second task, etc. The maximal value of k can be explicitly stated by rearranging the condition:

$$kT_i + D_i \leq t$$
$$\iff 0 \leq k \leq \lfloor \frac{t - D_i}{T_i} \rfloor$$

The highest value is $k = \lfloor \frac{t-D_i}{T_i} \rfloor$. If this value is negative, it means that no job of task τ_i respects the condition, i.e. $h_i(t) = 0$. Else, we can inject it directly and obtain $h_i(t) = \lfloor \frac{t-D_i}{T_i} \rfloor + 1$

This is generalized by the final definition of $h_i(t)$:

$$h_i(t) = \max\{0, \lfloor \frac{t - D_i}{T_i} \rfloor + 1\}$$

With this result, a closed form of the DBF function in the synchronous case is obtained

$$\mathrm{dbf}(t) = \sum_{i=1}^{n} \max\{0, \lfloor \frac{t - D_i}{T_i} \rfloor + 1\}\, C_i$$

2.2 Feasibility test for EDF

In [2], the authors show that the DBF can be used to derive the following feasibility condition for EDF[1] on sporadic task sets. An adapted proof is given here.

Theorem 1.

$$(\tau_1, ..., \tau_n) \textit{ is feasible} \iff \mathrm{dbf}(t) \leq t\ \forall t$$

Proof. We first show that the condition is *necessary*. Indeed, if the condition is not fulfilled (i.e. $\exists t^*\ s.t.\ \mathrm{dbf}(t^*) > t^*$), it means that in an interval of t^* time units, the total execution time of the jobs whose arrival and deadline occurs within the interval is superior to t^*. The system is thus not feasible if the condition is not fulfilled.

Now, we prove that the condition is *sufficient*. For that, suppose that $\mathrm{dbf}(t) \leq t\ \forall t$.

... (The idea is to look at each interval beginning at an arrival time and finishing at a deadline (not necessarily of the same job). Then we can prove that if there is only those two tasks, they can execute. If there are other tasks, we can look at a greater interval in which they can all execute)

□

Furthermore, the authors show that in the synchronous case, it is sufficient to consider intervals of the form $[0, t_d]$, where the t_d correspond to job deadlines (i.e. the increases the demand-bound function) in the interval $[D_{min}, H]$.

For constrained deadline set, it is sufficient to consider intervals $[0, t_d]$ with $t_d \leq L$, where L is the first busy period and is the solution to the following recursive equation

$$L = \sum_{i=1}^{n} \lceil \frac{L}{T_i} \rceil C_i$$

which can be computed through fixed-point iteration

$$w_0 = \sum_{i=1}^{n} C_i$$

$$w_{k+1} = \sum_{i=1}^{n} \lceil \frac{w_k}{T_i} \rceil C_i$$

Even so, this test requires to check a number of instant which is exponential in the number of tasks, making the general determination of feasibility a NP-hard problem.

[1]And thus, EDF being optimal, for all optimal schedulers

3 Definitive idle time approach

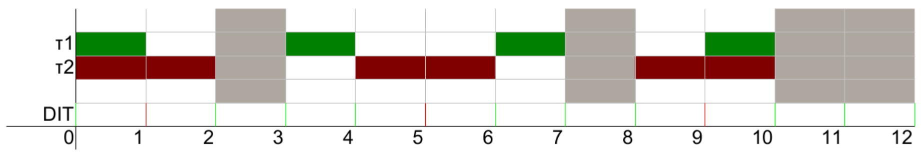
Definition 2. *An **idle time** is a time t such that every job released strictly before instant t finishes its execution before or at instant t.*

Definition 3. *A **definitive idle time** (DIT) is a time t such that every job released strictly before instant t has its absolute deadline before or at instant t.*

Remarks

- If the system is feasible, every DIT is an idle time.
- Contrary to the idle times, the DITs are independent of the scheduling or the execution times of the jobs.
- $t = 0$ is always a DIT.
- There are no DITs strictly after the first arrival of an arbitrary deadline task. In the following we only consider the constrained deadline case.

An example of the DITs of a system is given in Fig. 1.



	O	T	D	C
1	0	3	1	?
2	0	4	2	?

Figure 1: The block of colors represents the arrival time and deadline of a job. DITs are represented by a green line below

While the definition is applicable to both synchronous and asynchronous tasks system, the following only consider the synchronous case (which is the worst-case for non-concrete tasks). The difficulties arising with asynchronous tasks are described in a later section.

In the synchronous case, $t = 0$ and $t = H$ are DITs but there might be others in-between. The following notion is thus defined:

Definition 4. *The earliest DIT occurring in the system strictly after $t = 0$ is called the **first DIT** and is noted t_d.*

If the first DIT is H, then the only DITs of the system are multiple of H as the occurrence of DITs is periodic.

The following section explains the methods to determine the value of the first DIT. Then section 3.2 will explain a method to reduce the complexity of the DBF feasibility test if the first DIT occurs before H.

3.1 Existence and value

In synchronous systems, H is always a DIT, so the first DIT t_d exists and is such that $t_d \leq H$. In the constrained deadline case ($D_i \leq T_i \ \forall i$), it is possible that other DITs occur before it, in which case $t_d < H$.

The problem of the existence of a DIT can be written with modular equations as

$$\exists ?\ t_d\ s.t.\ \forall i : \exists a_i \in [D_i, T_i]\ s.t.\ t_d \equiv a_i\ (mod\ T_i)$$

For fixed values of a_i, this is a case of the Generalized Chinese Remainder Theorem (see Appendix A). However, in our case, each a_i can take every value in the interval $[D_i, T_i]$. Observe that the particular case $a_1 \cdots a_n = 0 \cdots 0$ is always a solution, which confirm the previous result that both 0 and H are always DIT.

If the value returned by the Chinese Remainder Theorem with a particular set of a_i values is called $t_{idle}(a_1, \cdots, a_n)$[2], the first DIT is given by

$$t_d = \min_{a_i \in [D_i, T_i],\ i=1\cdots n} t_{idle}(a_1, \cdots, a_n)$$

which require to consider a number of combinations which is exponential in the number of tasks.

3.2 Feasibility test

Theorem 2. *Let t_1 be a DIT, it is sufficient to check the DBF condition at job deadlines in the interval $[0, t_1]$ to determine feasibility.*

Proof. Let t_1 be any DIT of a synchronous task set such that $\forall t \leq t_1 : \mathrm{dbf}(t) \leq t$. We want to show that $\forall t > t_1 : \mathrm{dbf}(t) \leq t$.

As we are in the synchronous case, $t = 0$ is a critical instant, which mean we have the earliest next job deadline for each task. Thus the DBF increase faster than in any other configuration of arrival times.

A DIT for some task set is similar (w.r.t. the increase of the DBF) to the initial time of the same system in the asynchronous case. In other words, after the DIT, the increase of the DBF will be slower than it was at the initial time. Which gives us, with t_1 being the DIT:

$$\forall t > t_1 : \mathrm{dbf}(t) \leq \mathrm{dbf}(t_1) + \mathrm{dbf}(t - t_1)$$

We now prove by induction that $\mathrm{dbf}(t - t_1) \leq t - t_1$.

Initial case : We consider $t = t_1 + 1$. Thus, $\mathrm{dbf}(t - t_1) = \mathrm{dbf}(t_1 + 1 - t_1) = \mathrm{dbf}(1)$. As $1 \leq t_1$, we know that $\mathrm{dbf}(1) \leq 1$.

[2]If the values are such that there are no solution, we set t_{idle} to ∞

Inductive case : We consider any $t > t_1 + 1$ and we suppose that $\forall t' < t : \mathrm{dbf}(t') \leq t'$. Then, we have $\mathrm{dbf}(t - t_1) \leq t - t_1$ as $t - t_1 < t$.

And thus we have that $\mathrm{dbf}(t - t_1) \leq t - t_1$. As we also have $\mathrm{dbf}(t_1) \leq t_1$, the previous inequality becomes

$$\forall t > t_1 : \mathrm{dbf}(t) \leq \mathrm{dbf}(t_1) + \mathrm{dbf}(t - t_1) \leq t_1 + t - t_1 = t$$

$$\iff \forall t > t_1 : \mathrm{dbf}(t) \leq t$$

□

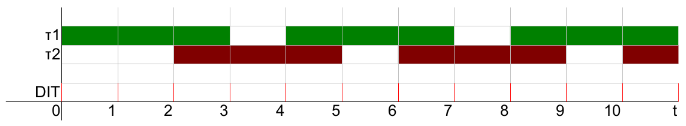
If there exists a DIT earlier than H, it can be used to reduce the interval used in the DBF feasibility test.

3.3 Results in the asynchronous case

Definition 5. *The* ***first periodic DIT*** *is the earliest DIT to occur strictly after the arrival of at least one job of each task.*

Note that the first periodic DIT will be the first DIT in the synchronous case.

In the synchronous case, $t = H$ is always a DIT. This is no longer true in the asynchronous case. In fact, the first periodic DIT could happen at a time $t_d > H$, or there could be no first periodic DIT at all, as seen in Fig. 2.



	O	T	D	C
1	0	4	3	?
2	2	4	3	?

Figure 2: An asynchronous task system with no DITs after $t = 0$

3.3.1 Existence and value

The first periodic DIT cannot happen at a time $t > O_{max} + H$, or else it would have happened previously.

Similarly to the synchronous case, the problem of existence of the first periodic DIT can be written as a system of modular equation:

$$\exists ?\ t_d\ s.t.\ \forall i\ :\\ \begin{cases} t_d > O_i \\ t_d - O_i \equiv a_i\ (mod\ T_i) \\ a_i \in [D_i, T_i] \end{cases}$$

which can, again, be solved by testing every combination of the a_i and solving the corresponding equation with the Generalized Chinese Remainder Theorem. If a solution $t < O_{max}$ is found, recall that $t + k\,H, k \in \mathbb{N}$ is also a solution. Note that this time, it is possible for the system to not have any solution (e.g. Fig 2).

3.3.2 Feasibility test

To understand feasibility tests in the asynchronous case, one must first understand how an asynchronous system behaves under, for example, EDF.

	Incomplete period	Transitive period	Stationary period	2^{nd} Stationary period	$\cdots$
t	0	O_{max}	$O_{max}+H$	$O_{max}+2H$	$\cdots$

Figure 3: Behavior of an asynchronous system under EDF. The value of t indicates the time at which each period starts

As shown in Fig. 3, the system begins in the *incomplete period*, where all tasks are not in the system. Then, after O_{max} time units, every tasks is in the system and the *transitive period* begins. Then, after $O_{max}+H$ time units, the system is faced with the same pattern of arrival as in O_{max}. However, because in $O_{max}+H$ every task is in the system and in O_{max} some were missing, the state of the system will be different. The *stationary period* then begins. It is only after $O_{max}+2H$ time units that the system comes back into a state in which it was before.

For this reason, as explained in [6], most feasibility test for asynchronous task system (e.g. the DBF test) are restricted to the interval $[0, O_{max}+2H]$, which has an exponential length in the number of tasks.

However, using DIT, we have the following property:

Theorem 3. *Let t_d be the first periodic DIT of an asynchronous system τ (supposing it exists). Then it is sufficient to check the DBF test in the interval* $[0, t_d+H]$

Proof. ???

(basically at t_d+H we know that the behavior will be the same as in t_d) □

4 Simulation

As was shown previously, the DBF condition must be true on all instant to prove that a system is feasible. It is necessary and sufficient to check only instants where a deadline occurs, and it is also necessary and sufficient to check only instants before the hyper-period, the end of the first busy period (for constrained system only), or the first DIT. While the test itself will be shorter if the smallest of those values is used as an upper limit, the time needed to compute the value must be taken into consideration as well.

The DBF feasibility test was simulated on a set of synchronous system with constrained deadline. For all of those systems, the value of the hyper-period, the end of the first busy period and the first DIT is computed. Then the DBF feasibility test is applied using those upper limits. The times each methods took to complete are compared.

4.1 Implementation model

4.1.1 Tasks system generation

The method used to generate the system consisted of

- Generating the U_i with the `UUniSort` algorithm described in [3], with parameters
  - U_{tot} (system utilization) chosen uniformly between 0.25 and 0.75
  - n (number of tasks) chosen uniformly between 1 and 4
- Generating the T_i from a list of divisors (see 4.1.2). This also gives the $C_i = \lfloor U_i T_i \rfloor$
- O_i (...)
- And finally, choosing the D_i uniformly in the interval $[T_i - \frac{T_i - C_i}{e}, T_i)$, where the value of e is discussed in 4.1.2.

4.1.2 General remarks

- The interval used for the generation of D_i can be explained by looking at the final formula for the DIT

  $$t_d = \min_{a_i \in [D_i, T_i],\ i=1\cdots n} t_{idle}(a_1, \cdots, a_n)$$

  The necessary number of combination of a_i values to considerr is equal to the cardinality of $[D_1, T_1] \times [D_2, T_2] \times \cdots \times [D_n, T_n]$, which gives an exponential time in the number of tasks. This constraint aims to allow the DIT algorithm to terminate in an acceptable time but it also gives it a considerable advantage.
- The T_i are not generated uniformly, but from a list of divisors of 554400[3]. This prevents the value of the hyper-period from growing exponentially. Similarly, this is giving the hyper-period method an advantage.
- During the DBF test, the deadline are considered chronologically. The order is not important for feasible tasks set, but can change the duration of the test for unfeasible task set (where the DBF test can stop when it finds an instant where the test does not hold).

4.1.3 Source code

Am I allowed to make it available on GitHub? License problem with ECE?

[3]LCM of 2^5, 3^2, 5^2, 7 and 11

4.2 Results

Fig. 4 gives us the precise timing with a value of $e = 5$. While the DIT and the hyper-period methods achieve similar results, the busy period method is clearly ahead. This can be explained by the relative simplicity of its algorithm.

Looking more closely at the values, the trade-off offered by the DIT method compared to the hyper-period method becomes apparent. While the preprocessing time is longer for the DIT, the interval to consider for the DBF test is greatly reduced.

```
== Test Results (on 1000 tasks system)
  Algorithms performance (upper limit computation + dbf test)
    Time with busy period: 0.02 + 0.01  =  0.03 s
    Time with DIT: 0.84 + 0.83  =  1.67 s
    Time with hyperT: 0.02 + 1.64  =  1.66 s
  Feasible? 432 , or about 43 %
```

Figure 4: Comparison of performance of different upper limit for the DBF feasibility test
with $e = 5$

Fig. 5 shows the influence of the e parameter, which describes the restriction imposed on the deadline parameter. Observe the exponential growth to the left. While we could think that the hyper-period curve should not be affected by e, this figure shows otherwise. This is explained by the fact that higher value of e means more feasibility, and as was shown previously more feasibility means longer feasibility tests.

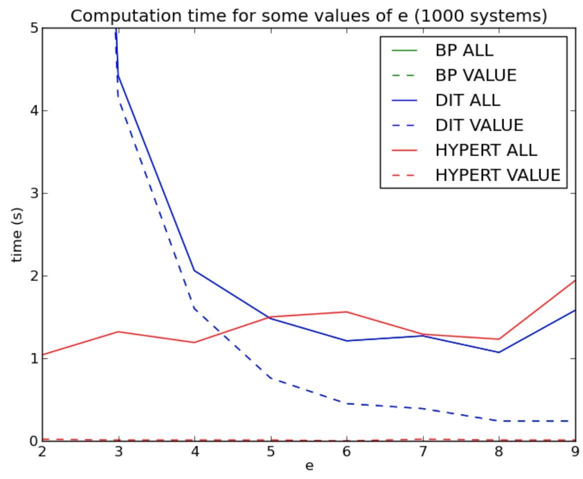


Figure 5: Comparison of performance for different values of e. The plain line represent the total time, while the dotted line represent only the upper limit computation time.

APPENDIX

A Generalized Chinese Remainder Theorem

First let us recall the Chinese Remainder Theorem, proven in [5].

Theorem 4. ***Chinese Remainder Theorem*** *Suppose $M_1, \cdots, M_n$ are positive integers which are pairwise co-prime. Then, for any given sequence of integers $a_1, \cdots, a_n$ there exists one integer x modulo $M = M_1 \cdots M_n$ solving the following system*

$$\begin{cases} x \equiv a_1 \ (mod\ M_1) \\ x \equiv a_2 \ (mod\ M_2) \\ \vdots \\ x \equiv a_n \ (mod\ M_n) \end{cases}$$

This value is given by

$$x = \sum_{i=1}^{n} a_i \frac{M}{M_i} \left[\frac{M}{M_i}\right]_{M_i}^{-1}$$

where $[a]_b^{-1}$ is the modular multiplicative inverse of a (mod b)

We can see that a requirement of the theorem is that the values of M_i need to be pairwise co-prime. When this is not the case, the system is not guaranteed to have a solution. Indeed, a previous result in [5] is that the system has one solution modulo M if and only if $\forall\ i, j\ s.t.\ a_i \equiv a_j\ (mod\ gcd(M_i, M_j))$ (which is clearly true when all the M_i are pairwise co-prime).

A method to find this solution, which we will call the Generalized Chinese Remainder Theorem, is given in [8]. The method consists of transforming the system into an equivalent one where the moduli are pairwise co-prime, which we can then solve using the Chinese Remainder Theorem. Another method would be the method of successive substitution.

References

[1] Sanjoy Baruah, Deji Chen, Sergey Gorinsky, and Aloysius Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, 1999.

[2] Sanjoy K Baruah, Louis E Rosier, and Rodney R Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990.

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

[4] Laurent George and Jean-François Hermant. Characterization of the space of feasible worst-case execution times for earliest-deadline-first scheduling. *Journal of Aerospace Computing, Information, and Communication*, 6(11):604–623, 2009.

[5] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen. *Introduction to algorithms*. The MIT press, 2001. Section 31.5.

[6] Joseph Y-T Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4):237–250, 1982.

[7] Giuseppe Lipari, Laurent George, Enrico Bini, and Marko Bertogna. On the average complexity of the processor demand analysis for earliest deadline scheduling. In *Proceedings of a conference organized in celebration of Professor Alan Burns' sixtieth birthday*, page 75.

[8] Arturo Magidin. Chinese remainder theorem with non-pairwise coprime moduli. `http://math.stackexchange.com/questions/120070/`, 2010. Accessed: 2013-07-16.