Periodic real-time scheduling: from deadline-based model to latency-based model

*Liliana Cucu(1), Nicolas Pernet (2) and Yves Sorel(3)**
(1) LORIA-INPL France, TRIO team
615 rue du Jardin Botanique
54600, Villers les Nancy
liliana.cucu@loria.fr, +(33) 3 54 95 84 62
(2) INRIA Rocquencourt France, AOSTE team
Domaine de Voluceau
78153, Le Chesnay
nicolas.pernet@inria.fr, +(33) 1 39 63 55 80
(3) INRIA Rocquencourt France, AOSTE team
Domaine de Voluceau
78153, Le Chesnay
yves.sorel@inria.fr, +(33) 1 39 63 52 60
Corresponding author: Liliana Cucu

*These results were obtained while the first author was at INRIA Rocquencourt

Abstract

This paper presents a connection between two real-time models: a deadline-based model and a latency-based model. The importance of the latency-based model is proved through a result showing that two deadlines, instead of a latency constraint, over-constrain the real-time applications. Moreover, we give a deadline-marking algorithm based on the relation between deadlines and latency constraints. This algorithm provides non-preemptive feasible schedules for systems with precedence constraints and deadlines, or more complex systems with deadlines and latencies. This is the first step toward non-preemptive schedulability for distributed architectures (without over-constraining the system) like, for example, the automotive applications using protocols such as Controller Area Network (CAN).
Keywords: scheduling, real-time, non-preemptive, latency, deadline, periodicity.

Real-time systems are, first of all, reactive systems, i.e., an output event produced by actuators reacting to an event coming into the system through sensors. This behavior is commonly seen in periodic systems of operations. In the classical deadline-based model given in Liu (1976), an operation becomes available at its release time which is repeated periodically. The start time of an operation is greater or equal to its release time. The operation must be scheduled before its deadline which is relative to the release time. The formal definition of this model is given in section 1. We refer to the classical model as the "deadline-based model".

However, the deadline-based model with release times is not adaptable to some applications such as real-time control. For example, if the start times of input and output operations of a control law are not periodic, then the performances of the controller decrease. In Torngren (1998), the author lists different results which take into account this variable delay by increasing the complexity of the control law, but the delay between control law computation and the output operation need to be constant. Therefore, strict periodicity constraints are necessary in these cases (see Korst (1996)).

In our model given in Cucu and Sorel (2002) besides the strict periodicity constraints, two other types of constraints may be imposed on these systems. Firstly, since some operations can produce data consumed by other operation(s), we need to specify that a producer operation must be executed before that of the consumer. So we impose precedence constraints between these operations, e.g., by specifying the application through a directed acyclic graph. Secondly, in some applications, we need to bound the delay between the start time of an operation and the end of another operation. For example, we consider a system which locates a flying object and has to estimate its trajectory every 5 ms. If three different sensors working with a 5 ms periodicity acquire the x, y and z space positions of the object, then several seconds might elapse between the x acquisition and the y. Such a localization method leads to a very bad trajectory estimation. One solution is to bound the delay between the first acquisition and the last. In these specific cases, this constraint is a *latency* constraint.

Theoretical definitions for this type of constraint were given in the case of data-flow graphs (see Goddard (2000)) and in the case of repeated precedence constraints (see van Beek and Wilken (2001)). These definitions are limited to particular cases and can not be employed to describe the elapsed time between any two operations related to each other by precedence constraints. Klein et al. also proposed modifications of the deadline, taking into account this type of constraint (see Klein et al. (1994)), but this over-constrains the system as we will prove in Section 1.

In Gerber et al. (1995) the authors justified the interest in "end-to-end" constraints, but the examples only contain "end-to-end" constraints between sensors and actuators. Obviously the latency constraint is an "end-to-end" constraint. Moreover, the latency constraints are not limited to sensors and actuators.

Our paper generalizes the results concerning end-to-end constraints (Gerber et al. (1995), Kang et al. (1997)). We take into account several latency constraints which are not necessarily imposed between one sensor and one actuator. In the localization example described previously, latency constraints are imposed between the three sensors (x, y and z) and if the aim of the trajectory estimation is to shoot the flying object, then another latency constraint can be imposed between the computation of the estimated next position of the flying object and the resulting shot. So, in this application there are multiple latency constraints but none of them is imposed between sensors and actuators.

We give the formal definition of our model in Section 1. We will refer to our model as

"latency-based model".

The results for a non-preemptive deadline-based model with release times, periodicity constraints and deadlines are poor (see Jeffay et al. (1991)). The interest in these systems is growing, e.g., in automotive applications using protocols like CAN. This paper presents a connection between the deadline-based and the latency-based models. Therefore, using the latency-based problem, we propose new results for the deadline-based problem and for more complex systems containing both models that can be found in distributed systems.

This paper is composed of four sections. The following section introduces the two models. Section 2 gives an algorithm for choosing the next operation to be scheduled, among the available operations, relative to the deadlines of their successors. Section 3 proposes an algorithm transforming an instance of a scheduling problem using the deadline-based model into an instance of a scheduling problem using the latency-based model. The paper ends with Section 4 giving suggestions for future research.

1 Two real-time models

Before presenting the two models, we give the following definitions (by system of operations we understand a set of operations with computation times and different constraints):

Definition 1 *For a system of operations, schedule S is a total order on the set of operations associating each operation A, a start time s_A which is the time instant when the operation starts its execution. We denote by $\mathcal{S}$ the set of the possible schedules of a system of operations. An operation is scheduled once its start time is known.*

Definition 2 *A system of operations is schedulable if there is at least one schedule satisfying the constraints of the system.*

We present the deadline-based model given by Liu and Layland (see Figure 1) in the non-preemptive case. An operation A becomes available, i.e., it requests its execution, at its release time r_A. The release time is periodically repeated with a period T_A. So if the first release of an operation A is $r_0 = r_A$ then the i'th release time of the operation is $r_i = r_A + iT_A, \forall i \geq 1$. Each instance of an operation A which requests its execution at $t = r_i$ must finish its execution by $r_i + D_A$, where D_A is the deadline of A. We denote by C_A its computation time and we consider worst-case execution times.

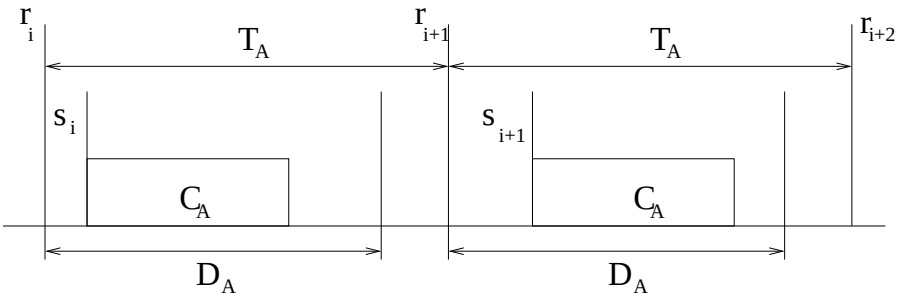

Figure 1: Model proposed by Liu and Layland

For the latency-based model, the operations are always executed as soon as they are available and they are repeated periodically with a period T_A (see Figure 2). So if the first start time $s_0 = s_A$ of an operation A is known, then the i'th start time of the operation is

$s_i = s_A + iT_A, \forall i \geq 1$. The computation time C_A is also known and we consider worst-case execution times.

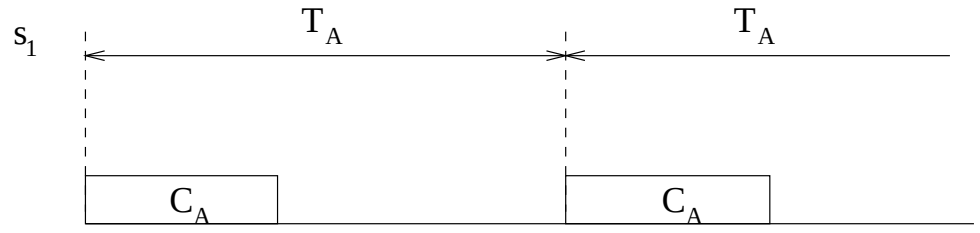

Figure 2: Model of periodic operations without release time

In the deadline-based model the release time of an operation is known, but not its start time. Moreover, for an operation the start times of all instances are not known at the moment when the start time of the first instance is.

In the latency-based model only the start time of the first instance of an operation is unknown. Once the start time of the first instance is known, the start times of all the instances are also known and their release times are useless for this model.

The difference between the two models is depicted for an operation A in Figure 3. We consider the deadline-based model for the first schedule and the latency-based model for the second.

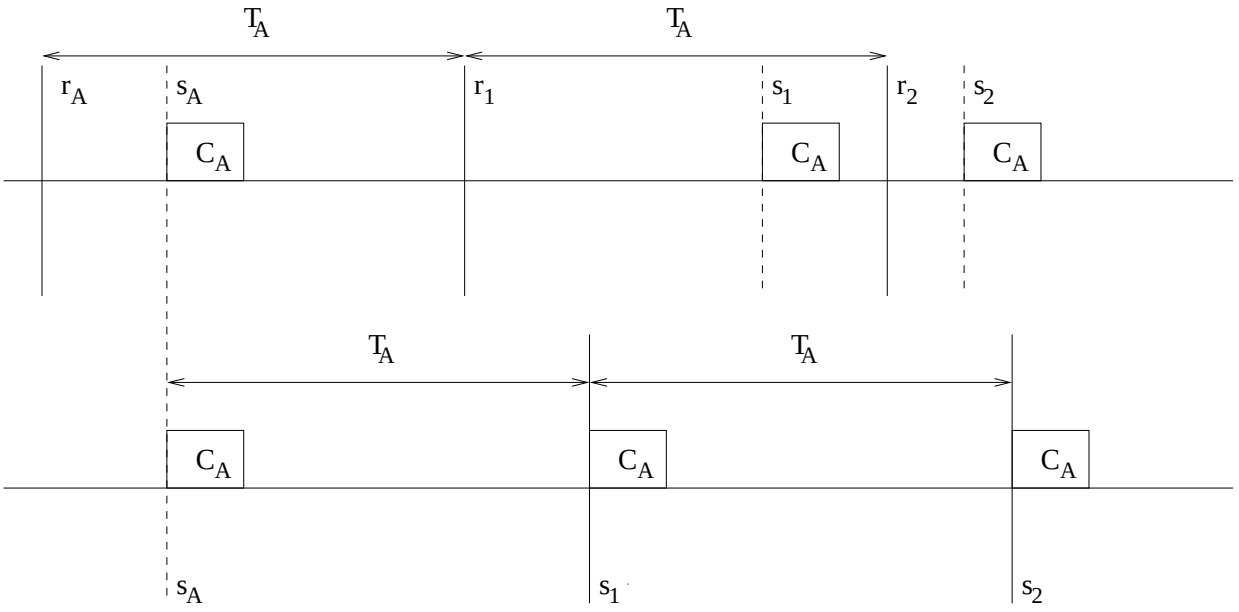

Figure 3: Difference between the deadline-based model (above) and the latency-based model

In the latency-based model we can define precedence constraints and latency constraints for the same set of operations. We use a graph-based model to specify the precedences between operations through a directed graph $G = (V, E)$ where V is the set of operations and $E \subseteq V \times V$ the set of edges which represents the precedence constraints between operations. Therefore, if the directed pair of operations $(A, B) \in E$, then B can be executed, only if A has already been executed. In the case of precedence constraints,

an operation is available when all its predecessors are already executed (see Cucu and Sorel (2003)).

Since the graph describes a real-time system, which is reactive, this latter graph has a *pattern* infinitely repeated (Grandpierre et al. (1999)) which induces an infinite repetition of all operations (see Figure 4).

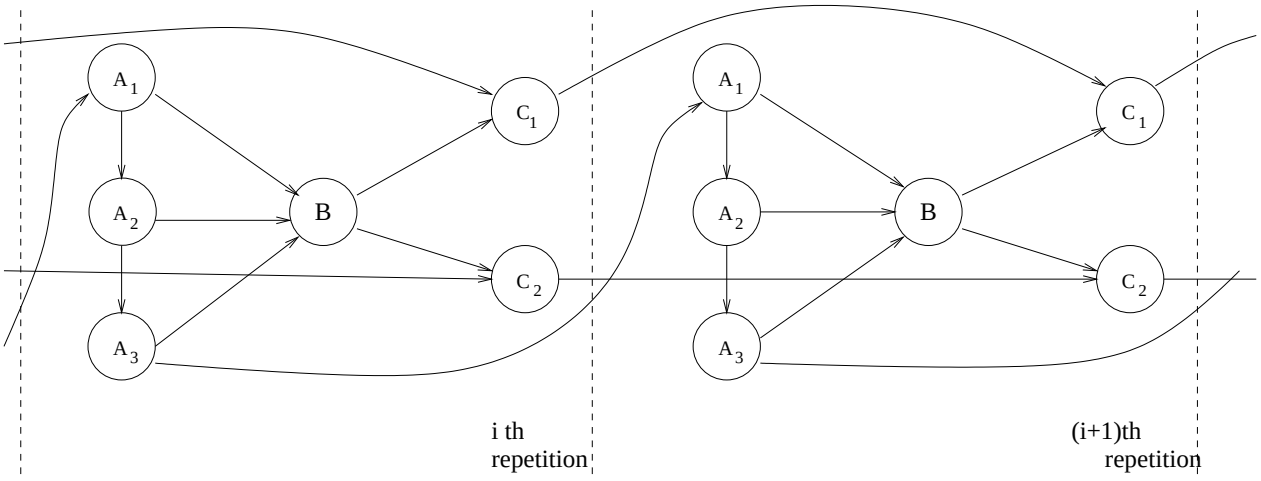

Figure 4: Graph describing a real-time system

For the sake of simplicity and since we use only directed graphs, we will use the term path instead of directed path.

We denote by $\mathcal{P}$ the set of all paths of graph $\mathcal{G}$. If there is at least one path from A to B, then we have $P(A,B) \in \mathcal{P}$. If $\nexists P(A,B) \in \mathcal{P}$, then there is no path from A to B. We denote by $\mathcal{D}(A) = \{B \in V$ such that $P(A,B) \in \mathcal{P}\}$.

Definition 3 *On a pair of operations (A,B) with $P(A,B) \in \mathcal{P}$ and A and B belonging to the same pattern, a latency constraint $L(A,B)$ is imposed if the operations are scheduled such that $s_B + C_B - s_A \leq L_{AB}$, $L_{AB} \in \mathbb{N}^+$. The latency constraint $L(A,B)$ has the value L_{AB}. Let $\mathcal{L}$ be the set of all latency constraints imposed for a system.*

The deadline can not describe a constraint between the start time of an operation and the completion time[1] of another operation as the latency constraint does (see Definition 3). If we want to describe this type of constraint using the deadline-based model, then we need to modify the deadline of the second operation relative to the parameters of the first operation as was done in Klein et al. (1994).

We consider the system of two operations $A = (r_A, C_A, T, D_A)$ and $B = (r_B, C_B, T, D_B)$ with the same period. We can impose the latency constraint $L(A,B)$ with the value L_{AB} by modifying the deadline of B. Lemma 1 gives this deadline modification and proves that the constraint we want to impose is verified. We give the obtained set in Figure 5.

Lemma 1 *We consider two operations $A = (r_A, C_A, T, D_A)$ and $B = (r_B, C_B, T, D_B)$ and a latency constraint $L(A,B)$ with a value L_{AB}. The constraint $L(A,B)$ is satisfied if the deadline of B is modified as follows:*

$$D_B^* = \min(D_B, (r_A + L_{AB} - r_B)) \tag{1}$$

[1]the completion time is the time instant where an operation ends its execution

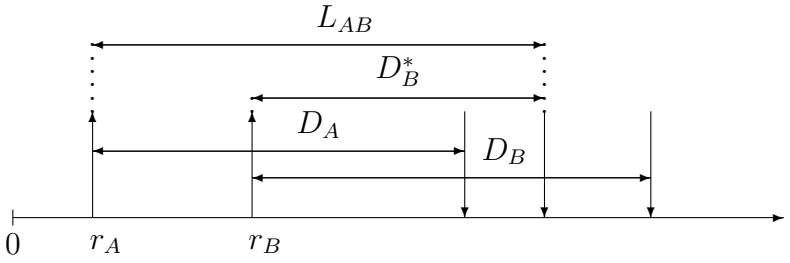

Figure 5: Modification of the operation B deadline to respect the latency constraint $L(A, B)$

Proof. We denote by s_A^i and s_B^i the start times of the i'th instance of A, respectively B. Similarly we denote by r_A^i and r_B^i the release times, and by f_A^i and f_B^i the completion times of the i'th instance of A, respectively B. We have $r_A^i = r_A + iT$.

We prove that if the deadlines of two consecutive instances of A and B are met, then the constraint $L(A, B)$ is satisfied, i.e., $\forall i\ s_B^i + C_B - s_A^i \leq L_{AB}$. Since C_B is constant, then $s_B^i + C_B - s_A^i$ is maximal if operation A starts as soon as possible, i.e., $s_A^i = r_A^i$, and operation B meets its deadlines but it starts as late as possible, i.e., $s_B^i + C_B = r_B^i + D_B$. Therefore we obtain:

$$\max_{k\in\mathbb{N}}\{s_B^k + C_B - s_A^k\} = \max_{k\in\mathbb{N}}\{r_B + k\,T + D_B^* - (r_A + k\,T)\} \leq r_B - r_A + D_B^*$$

From Equation (1) we have $\max_{k\in\mathbb{N}}\{s_B^k + C_B - s_A^k\} \leq r_B - r_A + (r_A + L_{AB} - r_B)$. We obtain $\max_{k\in\mathbb{N}}\{s_B^k + C_B - s_A^k\} \leq L_{AB}$ and the constraint $L(A, B)$ is satisfied. □

We consider the reciprocal implication of Lemma 1, i.e., a schedule that satisfies the constraint $L(A, B)$ will always satisfy the modified deadline. We prove below (see Lemma 2) that this implication is not true. It implies that the modification of the deadline reduces the space of solutions for a set of operations containing A and B. Thus the modification of the deadline over-constrains the set of operations.

Lemma 2 *For two operations $A = (r_A, C_A, T, D_A)$ and $B = (r_B, C_B, T, D_B)$ and a latency constraint $L(A, B)$ with a value L_{AB}, the deadline of B is modified as follows:*

$$D_B^* = \min(D_B, (r_A + L_{AB} - r_B)) \tag{2}$$

If the latency constraint $L(A, B)$ is satisfied then the modified deadline D_B^* is not necessary satisfied.

Proof. We consider the case where the i'th instance of the operation A starts its execution as late as possible and satisfies its deadline. This latest start time is $s_A^i = r_A^i + D_A - C_A$, and consequently, if the latency constraint $L(A, B)$ is satisfied, then the corresponding latest completion time for operation B is:

$$f_B^i = r_A^i + D_A - C_A + L_{AB}$$

$$f_B^i = r_A + iT + D_A - C_A + L_{AB}$$

We consider that the deadline of operation B was actually modified, i.e., $D_B > (r_A + L_{AB} - r_B)$. Consequently $D_B^* = r_A + L_{AB} - r_B$ and we have for i'th deadline of operation

B:

$$r_B^i + D_B^* = r_A + iT + L_{AB}$$

If we compare the latest completion time of operation B which satisfies $L(A, B)$ and its deadline, we obtain:

$$f_B^i - (r_B^i + D_B^*) = D_A - C_A$$

Consequently if $D_A > C_A$, then operation B can satisfy the latency constraint $L(A, B)$ without satisfying D_B^*. □

For an over-constrained set of operations it might be impossible to find a schedule that satisfies the latency constraint. For example for a set of three tasks A with $(0, 2, 10, 10)$, B with $(0, 3, 10, 10)$ and C with $(0, 8, 20, 20)$, we consider a latency constraint $L(A, B)$ of value 6. We obtain from operation B by modifying its deadline (see Equation (1)) a new operation B^* with $(0, 3, 6, 10)$. The set of operations $\{A, B^*, C\}$ is not schedulable whereas there is a feasible schedule for the set of operations $\{A, B, C\}$ given in Figure 6.

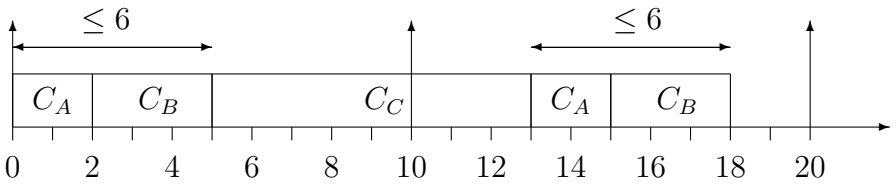

Figure 6: Example of schedule which satisfies the constraints of operation A, B and C

Without loss of generality we assume that all characteristics, i.e. the computation times, the periods and the latency constraints, are defined as multiples of a clock tick τ (time is discrete). Afterwards, the integer values of the given characteristics are implicitly multiplied by τ.

2 Deadline-marking algorithm

In this section, we present a theorem proving that a deadline in the deadline-based model can be expressed as latency constraint in the latency-based model. Thus we may solve the non-preemptive scheduling problem of systems with precedence, strict periodicity and deadlines. This problem is a sub-problem of latency-based problem with the latency constraints replaced by deadlines. The results of this section stand for both uniprocessor and multiprocessor cases.

The following theorem gives the relationship between the deadlines and the latency constraints.

Theorem 1 *A latency constraint $L(A, B)$ is equivalent to a deadline for the operation B, once the operation A is scheduled.*

Proof. We prove the theorem by double implication.

First, we prove that a latency constraint $L(A, B)$ of a pair $(A, B) \in \mathcal{L}$ may be expressed as the deadline of operation B, once A is scheduled. Since $(A, B) \in \mathcal{L}$, then we have $s_B \leq L_{AB} + s_A - C_B$. If we denote by D_B the expression $L_{AB} + s_A - C_B$, we obtain

$$s_B \leq D_B \tag{3}$$

Notice that in this case the deadline of B is relative to the time instant $t = 0$.

Once A is scheduled, s_A is known and D_B becomes known, also. Since s_A is calculated from the beginning of the schedule then the Equation (3) defines a deadline for B.

Second, we express the deadline of an operation B as a latency constraint $L(\cdot, B)$. Since B has a deadline D_B, then $s_B \leq D_B$. We denote by A the first operation which was scheduled, i.e., $s_A = 0$. Since the deadline is defined by the start of the schedule, we obtain $s_B - s_A \leq D_B$. Moreover, since the first scheduled operation is chosen among all the operations without predecessors, then we have $s_B - s_C \leq D_B, \forall C \in V$ with $Prec(C) = \emptyset$ and $B \in \mathcal{D}(C)$, where $Prec(C)$ is the set of predecessors of operation C. So, in order to satisfy the deadline of B, we define several latency constraints $L_{CB} = D_B$ for each C such that $Prec(C) = \emptyset$ and $B \in \mathcal{D}(C)$. □

Due to this theorem, we may use the scheduling algorithm given in Cucu and Sorel (2002) for systems with precedence, strict periodicity and latency constraints to schedule systems with precedence constraints, strict periodicity constraints and deadlines. This scheduling algorithm uses a latency-marking algorithm. Theorem 1 proves that we can modify the marking algorithm in order to take into account the deadlines. In this case the function $mark$ of operation A is equal to $min_{B \in \mathcal{D}(A)}\{D_B\}$. These marks are obtained using the following deadline-marking algorithm:

Algorithm 1

Initialization: $\mathcal{W} = \bigcup_{A \in V \text{ and } Suc(A)=\emptyset} Prec(A)$ is the working-set. If A has a deadline D_A, then $mark(A) = D_A$, otherwise $mark(A) = \infty$.

Step 1: for $A \in \mathcal{W}, mark(A) = min(mark(A), min_{B \in Suc(A)}\{mark(B)\})$ and $\mathcal{W} = \mathcal{W} \backslash \{A\}$. We add to $\mathcal{W}$ the operations for which all the predecessors has been removed from $\mathcal{W}$.

Step 2: repeat Step 1 until $\mathcal{W} = \emptyset$.

Remark 1 *Recursively, each operation inherits the deadlines of its successors. At the end of the deadline-marking algorithm, the mark of an operation may be equal to either its inherited deadline, or its initial deadline. Therefore, the algorithm does not modify the marks of the operations without successors. This algorithm is applied to the pattern.*

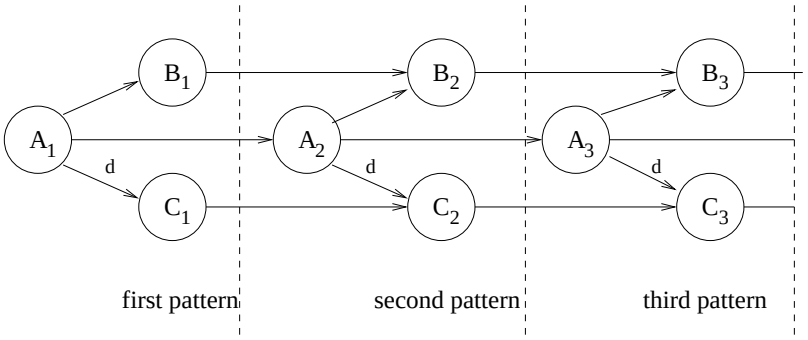

Figure 7: Graph of example 1

Example 1 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph given in Figure 7. We consider that only C_1 has a deadline D_{C_1}. In Figure 7, the letter d on the edge from A_1 to C_1 implies that the precedence constraint is due to data transfer. The algorithm is given in table 1, where by $m(A)$ we understand $mark(A)$.*

	$m(A_1)$	$m(A_2)$	$m(A_3)$	$m(B)$	$m(C_1)$	$m(C_2)$
Initialization	∞	∞	∞	∞	D_{C_1}	∞
D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}

Table 1: Marks obtained using the marking algorithm

3 Transformation from deadline-based problem to latency-based problem

In this section, we give an algorithm transforming the deadline-based model scheduling problem into a latency-based model scheduling problem. We prove that the two problems are equivalent from the point of view of complexity. Thus, the deadline-based problem may be solved in the non-preemptive case using the results for the latency-based problem. The results of this section stand for both uniprocessor and multiprocessor cases.

We consider a system with n operations and for each operation A, we have the computation time c_A, the release time r_A and the deadline D_A. The operation A is periodic with the period T_A. We transform this system using the following algorithm:

Algorithm 2

Step 1: for each operation A we add a fictive operation A' with the computation time equal to 0.

Step 2: we add a precedence constraint between A' and A. We obtain a graph of precedence constraints with $2n$ operations and n edges.

Step 3: we impose latency constraints for each (A', A) with $L_{A'A} = D_A$.

Step 4: we impose strict periodicity constraints for each operation A' with T'_A equal to the period of operation A.

Example 2 *This example illustrates Algorithm 2. For a system of three operations A, B and C with no precedence constraints, we obtain the graph given in Figure 8.*

We denote by PRE the decision problem associated to the scheduling problem of periodic systems with deadlines. We denote by $PPLc$ the decision problem associated to the scheduling problem built from PRE using the Algorithm 2.

Theorem 2 *The decision problem PRE has, as a solution, the answer "yes" if, and only if, the decision problem PPLc has, as a solution, the answer "yes".*

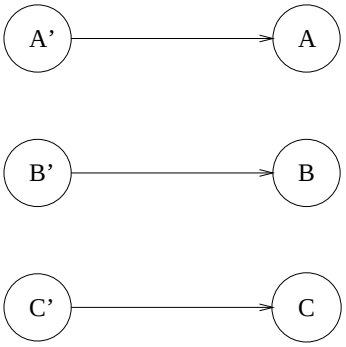

Figure 8: Graph obtained using the Algorithm 2

Proof. We prove the theorem by double implication.

For the decision problem PRE each operation A has a release time r_A, a deadline D_A defined from the release time, a period T_A and a computation time C_A.

We suppose that the problem PRE has the answer "yes" and we prove that the decision problem $PPLc$ also has the answer "yes". We denote by S one of the schedules which satisfies the constraints of scheduling problem PRE. This implies that for each operation A we know the start time $s_A \in S$ such that $r_A \leq s_A \leq D_A$.

From the schedule S, we build a schedule S' for the problem $PPLc$. Each operation A' associated to an operation A of an instance of the problem PRE has the start time of its first instance $s_{A'}$ equal to r_A. Each operation A of an instance of the problem $PPLc$ has a start time equal to the start time of the operation A in the schedule S. The Figure 9 illustrates this construction. The operation A' with the computation time equal to 0 is marked on the figure with non-continuous line. Consequently, since the deadlines are satisfied in the schedule S then all the latency constraints are satisfied in the schedule S'. Since the operations A' have computation times equal to 0 and their start times are periodically repeated due to the periodicity of release times in schedule S, then the periodicity constraints are satisfied in schedule S.

Similarly we build a schedule for problem PRE from a schedule of problem $PPLc$. Consequently, there is at least a schedule for the first problem if and only if there is a schedule for the problem built using the Algorithm2. The theorem is proved. □

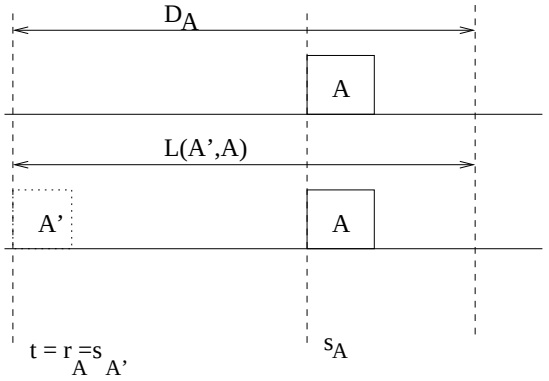

Figure 9: Construction of a schedule for $PPLc$ from a schedule for PRE

4 Conclusion and further research

We have presented two real-time models in the non-preemptive case. The first model is a deadline-based model and the second is a latency-based model. After comparing the two models and showing that there are applications that can not be described by deadlines without over-constraining the system, we prove that the deadline may be defined using a latency constraint.

The main contribution of the paper is that we provide a link between the deadline-based model and the existing schedulability results obtained for the latency-based model. Thus we provide an algorithm solving the scheduling problem with precedence, strict periodicity constraints and deadlines as a particular case of the latency-based problem. These results are the first step toward the schedulability analysis of systems such as distributed systems, where deadlines and latency constraints may coexist. Therefore as a future research project we propose to study these types of systems.

References

van Beek, P. and Wilken, K.D. (2001). "Fast Optimal Instruction Scheduling for Single-Issue Processors with Arbitrary Latencies", *Lecture Notes in Computer Science*, 2239, 625-629.

Cucu, L., Kocik, R. and Sorel, Y. (2002). "Real-time scheduling for systems with precedence, periodicity and latency constraints", Proceedings of *Real-time and Embedded Systems*, 173-188.

Cucu, L., and Sorel, Y. (2003). "Schedulability condition for systems with precedence and periodicity constraints without preemption", Proceedings of *Real-time and Embedded Systems*, 292-305.

Gerber, R., Pugh, W. and Saksena, M. (1997). "Parametric Dispatching of Hard Real-Time Tasks", *IEEE Trans. Computers*, 44(3), 471-479.

Goddard, S. (2000). "Constraints on data-flow", Ph.D. thesis from University of North Carolina.

Grandpierre, T., Lavarenne, C. and Sorel, Y. (1999). "Optimized Rapid Prototyping For Real Time Embedded Heterogeneous Multiprocessors", Proceedings of *Codes'99 7th International Workshop on Hardware/Software Co-Design*, 74-78.

Harel, D. and Pnueli, A. (1985). "On the Development of Reactive Systems", *Logics and Models of Concurrent Systems*, Spring-Verlag, 477-498.

Jeffay, K., Stanat, D.F. and Martel, C.U. (1991). "On Non-Preemptive Scheduling of Periodic and Sporadic Tasks", Proceedings of *IEEE Sympossium on Real-Time Systems*, 129-139.

Kang, D.-I., Gerber R. and Saksena, M. (1997). "Performance-Based Design of Distributed Real-Time Systems", Proceedings of *IEEE Real-Time Technology and Applications*, 2-13.

Klein, M.H., Lehoczky, J.P. and Rajkumar, R. (1994). "Rate-Monotonic Analysis for Real-Time Industrial Computing", *IEEE Computer*, 27(1), 24-33.

Korst, J.H.M., Aarts, E.H.L. and Lenstra, J.K. (1996). "Scheduling Periodic Tasks", *INFORMS Journal on Computing*, 9(4), 351-362.

Liu, C.L. and Layland, J.W. (1973). "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *Journal of the ACM*, 20(1), 46-61.

Torngren, M. (1998). "Fundamentals of implementing real-time control application in distributed computer systems", *Journal of Real-Time Systems*, 14, 219-260.