Fine-Tuning MAC-Level Protocols for Optimized Real-Time Quality-of-Service

Mathieu Grenier, Nicolas Navet

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fine-Tuning MAC-Level Protocols for Optimized Real-Time Quality-of-Service

Mathieu Grenier — Nicolas Navet

N° 6247

June 2007

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6247--FR+ENG

INRIA LORRAINE

Fine-Tuning MAC-Level Protocols for Optimized Real-Time Quality-of-Service

Mathieu Grenier * , Nicolas Navet *

Thème COM — Systèmes communicants
Projet TRIO

Rapport de recherche n° 6247 — June 2007 — 21 pages

Abstract: In distributed real-time systems, meeting the real-time constraints is mandatory but the satisfaction of other application-dependent criteria is most generally required as well. In particular, Networked Control Systems (NCS) are known to be sensitive to communication delays such as frame response time jitters. Well known Medium Access Control (MAC) algorithms such Non-Preemptive Deadline Monotonic (NP-DM) or Non-Preemptive Earliest Deadline First (NP-EDF) are efficient in terms of bandwidth usage but they may perform poorly regarding other application dependent performance criteria. This paper highlights a class of on-line scheduling policies targeted at scheduling frames at the MAC level, and provides a schedulability analysis that is valid for all policies within the considered class. As it will be shown, these algorithms are easily implementable on COTS components (e.g., Controller Area Network controllers) and offer good trade-offs between feasibility and the satisfaction of other application-dependent criteria such as the response time jitter.

Key-words: Real-time systems, control systems, communication systems, scheduling, protocols, field buses.

* LORIA - TRIO Group, Campus Scientifique - BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France. Email: First-Name.LastName@loria.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Optimisation de la qualité de service des protocoles MAC temps réel

Résumé : Dans les systèmes distribués temps réel, le respect des contraintes temps réel est *sine qua non* mais la satisfaction d'autres critères est également le plus souvent requis. En particulier, les "systèmes contrôlés en réseau" (*Networked Control System - NCS*) sont connus pour leur sensibilité aux délais de communication comme la gigue sur les temps de réponse. Les protocoles les plus courants pour l'accès au médium de communication (*Medium Access Control - MAC*) tel que Deadline Monotonic (NP-DM) ou Earliest Deadline First non-préemptif (NP-EDF) sont efficaces en termes d'utilisation de bande passante mais peuvent avoir un comportement médiocre au regard des autres critères de performances dépendants de l'application. Ce papier présente une classe de politiques d'ordonnancement en ligne pour l'ordonnancement des trames au niveau MAC, et fournit une analyse d'ordonnancement applicable à toutes les politiques appartenant à cette classe. Comme nous le montrerons, ces algorithmes sont facilement implémentables sur des composants du commerce, comme des contrôleurs de communication CAN, et offrent un bon compromis entre faisabilité et la satisfaction des critères propres à l'application comme la gigue sur les temps de réponse.

Mots-clés : Systèmes temps réel, systèmes de contrôle, systèmes de communication, ordonnancement, protocoles, réseaux de terrain.

1 Introduction

Context of paper In distributed real-time systems, the respect of timing constraint is a basic requirement but other application-dependent criteria besides feasibility are of interest. For instance, in Networked Control Systems (NCS), where the control system is distributed through a network, message exchanges induce non-constant delays in the control-loop, which affects the performance and even the stability of the controlled system [1, 2, 3]. The frame scheduling algorithm at the Medium Access Control (MAC) has a large influence on these delays and thus, should be chosen according to the application's performance requirements.

Problem definition The problem is to devise MAC level scheduling algorithms that are efficient in terms of bandwidth usage as well as for the satisfaction of application dependent criteria. Moreover, as far as possible, these algorithms should be implementable on COTS hardware and induce small overhead. In the following, we will consider that the MAC scheduling algorithm is implemented on top of a priority bus, such as CAN [4, 5], VAN [6] or J1850 [7]. Indeed, priority buses have proven to be effective in a real-time context, and are widely available and used.

Related work Many studies have been devoted to find scheduling solutions that are well suited to the application's requirements, both for the scheduling of tasks and the scheduling of messages.

1. Scheduling of tasks:

   In [8], a modified version of the Constant Bandwidth Server (CBS), initially proposed in [9], is used to eliminate jitters. Data input and data output occur at fixed points in time and control tasks run in a CBS, which is an abstraction of a dedicated CPU offering a chosen fraction of the original CPU, to ensure that control tasks finish before the output of the data.

   To better fit to the processing requirements of a control system, new task models have been conceived. In [10, 11], the elastic task model is proposed to handle overruns: task adapt their period at runtime in such a way as to keep the systems underloaded. In [12, 13], it is proposed that control tasks are subdivided in three different parts: sampling, computation, actuation. Sampling and actuation sub-tasks are assigned a high priority in order to reduce the jitters.

   Another solution is to adjust the parameters of the tasks to achieve the desired goals. In [14], the worst-case end-of-execution jitter is minimized by choosing appropriate deadlines. In [15], initial offsets and priorities are adjusted to reduce jitter by minimizing preemption.

   Improvements can also be brought by well choosing the parameters of the scheduling policies. In [16], a priority allocation scheme is proposed to reduce the average response time while, in [17], the problem of choosing scheduling policies and priorities on a Posix 1003.1b compliant operating system (OS) is tackled.

   Finally, another way is to create new scheduling policies. In [18], the scheduler is synthetized as a timed automata from the Petri net modeling the system and the properties expected from the system. In [19], also starting from a Petri net model of the system, an optimal scheduling sequence is found by examining the marking graph of the Petri net. In direct link with this study is the work published in [20], where on-line task scheduling policies are conceived for optimizing application-dependent performances criteria.

2. Message scheduling:

   Efficient scheduling schemes have been proposed in [21] and [22] to ensure fairness and bandwidth isolation between streams of messages sent on a CAN bus, respectively with a centralized (i.e., a master node has full control over the scheduling) and decentralized scheme.

   Other studies proposed solutions for implementing Non-Preemptive Earliest Deadline First (NP-EDF) in order to achieve a higher network utilization rate while meeting timing constraints. This is done on

CAN in [23, 24, 25] with solutions where deadlines are encoded on a logarithmic time scale, which has been shown to minimize the priority inversions due to the limited number of priority bits. In [26], a NP-EDF implementation is proposed on top of FTT-CAN while [27] addresses the same problem when the application makes use of the standardized CAN application layers CAL CiA [28] and CANOpen [29].

Another scheme, published in [30], allows to reduce the response time jitter due to bit-stuffing by introducing some restrictions on the priorities that can be allocated to the stream of frames on a CAN network. To achieve the same goal, the authors in [31, 32] propose a genetic algorithm to adjust the initial offsets of messages in order to reduce transmission jitter (same goal as [15] for the uniprocessor case).

Overview of the approach In this study, the problem addressed is to efficiently grant access to the network at the MAC level**.** The performance criteria are the schedulability of the systems and the satisfaction of application dependant criteria, such as the frame response time jitter which is crucial for Networked Control Systems. The proposal is comprised of three distinct steps:

1. define the properties that a "good" MAC-level real-time scheduling policies must possess and identify a sub-class, particularly promising in terms of performances,
2. derive a schedulability analysis for this sub-class class of scheduling policies and address the implementation issues,
3. explore the search space and find out the most efficient policies. Two cases can be distinguished. The case where the policy has to be efficient on average (it can be used with different traffic streams whose characteristics are a priori not known). And, the case where the policy should be fine tuned for a particular application.

Organisation of the paper In section 2, the computational model is stated. In section 3, the properties required from a "good" scheduling algorithm in the real-time context are exihibited. Among this set of "good" policies, the sub-class considered in the following is presented. In section 4, the implementation issues on top of the CAN priority bus are discussed. Section 5 is devoted to the schedulability analysis. Finally, experiments showing the effectiveness of the proposal are summarized in section 6.

2 Network model

This study deals with the scheduling of frames at the Medium Access Control level; in particular it is assumed that segmentation takes place in the upper layers of the communication stack. The access to the bus is granted by an algorithm (e.g., priority based, token bus, etc.) that is termed the *scheduling policy* in the rest of the paper.

2.1 Traffic stream model

A set $\mathcal{F}$ of m traffic streams is considered. Nodes, interconnected through a network, send the streams. There may be several traffic streams sent by the same node. Each traffic stream f_k generates and queues periodically frames, where $f_{k,n}$ denotes the n^{th} frame of the traffic stream f_k. By analogy to the periodic task model used in CPU scheduling, a traffic stream f_k is characterized by a triple $(C_k, \overline{D}_k, T_k)$ where C_k is the *worst-case transmission time* of a frame belonging to the stream, $\overline{D}_k$ the *relative deadline* (i.e. maximum tolerable delay between the transmission request and the reception at all receiving nodes), and T_k the *inter-arrival time* between two instances of f_k. The release time of a frame $f_{k,n}$ is denoted by $A_{k,n}$. For the sake of clarity, sporadic streams and jitters in the availability dates are not considered here but can be taken into account as classically done, see [33] for instance.

Concrete and non-concrete traffic streams A *concrete* traffic stream $(f_i, A_{i,1})$ is a stream for which the release time of its first frame $A_{i,1}$ is known before run-time while the release times of *non-concrete* streams are unknown. A set of n non-concrete traffic streams is denoted by $\Omega = \{f_1, f_2, ..., f_n\}$, where f_k is the stream with index k, while a set of n concrete traffic streams is $\omega = \{(f_1, A_{1,1}), (f_2, A_{2,1}).....(f_n, A_{n,1})\}$, where $A_{k,1}$ is the initial offset of stream f_k (i.e. the release time of the first instance). Without restrictions on the initial offsets, there is an infinite number of mapping from the set of non-concrete traffic streams Ω to the set of concrete traffic streams ω.

Feasibility and optimality A concrete set of traffic streams ω is said *feasible* (or *schedulable*) if no frame of the system is received after its absolute deadline $D_{k,n} = A_{k,n} + \overline{D_k}$. A non-concrete set of traffic streams Ω is feasible, if all the concrete sets ω, which can be generated from Ω, are feasible.

A scheduling policy is *optimal* with respect to a certain criterion (e.g. feasibility, average response time) within its class if no other policy of the class performs better with respect to the criterion. In the following, *optimal* is used to mean *optimal* with respect to feasibility. A scheduling policy is *non-concrete optimal* (with respect to feasibility) if it successfully schedules all the non-concrete sets that are schedulable with a policy of the class. As shown in [34], Non-Preemptive Earliest Deadline First (NP-EDF - smaller the absolute deadline, greater the priority) is non-concrete optimal within the class of non-idling policy, where non-idling means that the network is not idle when there are frames awaiting for transmission.

2.2 Defining policies through priority function.

Priority functions is a convenient way of formally defining in a non-ambiguous manner scheduling policies, which to our best knowledge, was introduced for the first time in [35]. The priority function $\Gamma_{k,n}(t)$ indicates the priority of a frame $f_{k,n}$ at time t . The network is allocated, at each time, according to the *Highest Priority First* (HPF) paradigm.

Function $\Gamma_{k,n}(t)$ takes its value from a totally ordered set $\mathcal{P}$, which is chosen in [35] to be the set of multi-dimensional $\mathbf{R}$-valued vectors $\mathcal{P} = \{(p_1, ..., p_n) \in \mathbb{R}^n \mid n \in \mathbb{N}\}$ provided with a lexicographical order. Between two vectors, coordinates are compared one by one starting from the left, the first different coordinate decides the priority order with the convention "the smaller the numerical value, the higher the priority". For instance, $\Gamma_{i,j}(t) = (3, 4, 5)$ and $\Gamma_{k,n}(t) = (3, 4, 6)$ implies that $f_{i,j}$ has a higher priority than $f_{k,n}$ at time t. Priority vectors of different sizes can be compared with the same rule as above and the convention that a missing coordinate is the lowest numerical value (e.g. $\Gamma_{i,j}(t) = (3, 4, 5)$ and $\Gamma_{k,n}(t) = (3, 4)$ means that $f_{k,n}$ has a higher priority than $f_{i,j}$ at time t). Finally, two vectors are equal iff they have the same size and if the components are equal one by one. As it will be explained in section 4, on a priority bus such as CAN, the priority vector can be mapped to the priority coded in the Identifier field of the frames. This allows to take advantage of the "native" arbitration mechanism of the priority bus to perform the arbitration according to the scheduling policy defined by the priority function $\Gamma_{k,n}(t)$.

Most real-time scheduling policies can be defined easily using priority functions. For instance, the priority of a frame $f_{k,n}$ under NP-EDF and Non-Preemptive Deadline Monotonic (NP-DM - smaller the relative deadline, greater the priority), priority becomes:

$$\Gamma_{k,n}^{NP-EDF}(t) = \begin{cases} (A_{k,n} + \overline{D}_k, k, n) \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases} , \Gamma_{k,n}^{NP-DM}(t) = \begin{cases} (\overline{D}_k, k, n) \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

where $B_{k,n}$ is the transmission begin of $f_{k,n}$ (the last two coordinates k, n are needed to ensure decidability, see definition 2). The $-\infty$ value of the first component of the priority vector after the start of transmission (i.e. $t \geq B_{k,n}$) ensures non-preemptiveness. A class of policies of particular interest, to which NP-EDF and NP-DM belong, is the *Priority Promotion at Execution Beginning* policies.

Definition 1 *[35] A scheduling policy $\mathcal{A}$ is a Priority Promotion at Execution Beginning (PPEB) policy if the priority of a frame may change only at the start of transmission $B_{k,n}$ of the frame:*

$$\Gamma_{k,n}^{PPEB}(t) = \begin{cases} \overrightarrow{P}_{k,n} \text{ if } t < B_{k,n} \\ \overrightarrow{Q}_{k,n} \text{ if } t \geq B_{k,n} \end{cases},$$

where the priority vector of a frame $f_{k,n}$ before (resp. after) its execution beginning is $\overrightarrow{P}_{k,n}$ (resp. $\overrightarrow{Q}_{k,n}$).

Besides providing non-ambiguous definition of the scheduling policy, priority functions enable us to distinguish classes of scheduling policies and to derive generic results that are valid for all the policies within a certain class. The next section presents the sub-class of non-preemptive scheduling policies that will be studied in the rest of the paper.

3 Study Domain

An arbitrary priority function does not necessarily define a scheduling of interest for real-time computing nor a policy that can be implemented in practice. In this section, the properties require from a "good" scheduling policy is precised, where "good" means a policy that can be implemented in practice and that is suited to real-time computing. Then, among the set of all good policies, the particular class of scheduling policies considered in this study is defined.

3.1 "Good" scheduling policies

A "good" policy must meet a certain number of criteria, which are needed for the policy to be implemented in a real-time context.

Decidable policies Policies must be *decidable*: at any time t, there is exactly one frame of maximal priority among the set of active frames (*i.e.* frames which contend for the channel). This concept of decidability was introduced in [35] for the uniprocessor case.

Definition 2 *[35] A priority function is decidable iff, at each time t such that work is pending, there is exactly one frame with the highest priority among the set of pending frames.*

For instance, the last two components of $\overrightarrow{P}_{k,n}^{NP-EDF} = (A_{k,n} + \overline{D}_k, k, n)$ ensure decidability in case of equal deadlines. Note that this property is also required for an implementation on a priority bus.

"Temporal invariant" policies In this study, for the sake of predictability of the system, the only scheduling policies of interest are such that the relative priority between two frames does not depend on the numerical value of the local clocks. The relative priority must remain the same if the arrival of all frames is "shifted"to the left or the right. The policy is thus independent of the value of the local clocks at start-up time.

Such policies are said *shift temporal invariant* (STI) policies. NP-EDF is a STI policy since the priority between two frames only depends on the offset between arrival dates and on relative deadlines. On the contrary, a policy defined by $\overrightarrow{P}_{k,n} = (C_k \cdot A_{k,n}, k, n)$ is not STI; just consider $f_{i,1}$ and $f_{j,1}$ with $A_{i,1} = 0$, $C_i = 10$ and $A_{j,1} = 1$ with $C_j = 1$ and the same two frames except that the arrival dates are shifted to the right by one unit of time.

Definition 3 *Let two concrete traffic stream sets be $\omega = \{(f_1, A_{1,1}), (f_2, A_{2,1})\ldots\ldots(f_n, A_{n,1})\}$ and $\omega' = \{(f_1, A_{1,1}+\Phi), (f_2, A_{2,1}+\Phi)\ldots\ldots(f_n, A_{n,1}+\Phi)\}$ where ω' is a "shifted" version of ω (with $\Phi \in \mathbb{Z}$).*

A scheduling policy $\mathcal{A}$ is Shift Temporal Invariant (STI) iff for all possible $\Phi \in \mathbb{Z}$, $\forall i, j, k, n$ such that $(k, n) \neq (i, j)$ (two distinct frames), one has:

$$\begin{array}{rll} \forall t\, \Gamma_{k,n}^{\mathcal{A},\omega}(t) & \succ (resp \prec) & \Gamma_{i,j}^{\mathcal{A},\omega}(t) \implies \\ \Gamma_{k,n}^{\mathcal{A},\omega'}(t+\Phi) & \succ (resp \prec) & \Gamma_{i,j}^{\mathcal{A},\omega'}(t+\Phi) \end{array}$$

where $\Gamma_{k,n}^{\mathcal{A},\omega}(t)$ is the priority of $f_{k,n}$ of the concrete traffic stream set ω at time t.

Implementable policies For being *implementable* in practice, components of the priority vector must be representable by a limited number of bits (for instance, 11 bits on standard CAN). In the following, coordinates of the priority vector take their value in the set of rational numbers $\mathbb{Q}$; the imprecision due to the limited number of available bits, called "quantization error" as in [23], will be taken into account in the schedulability analysis developed in paragraph 5.

Any "good" MAC-level scheduling algorithm intended to be implemented on top of a priority bus has to fulfill the aforementioned properties: the policy must be decidable, shift temporal invariant and implementable. In the next paragraph, the class of non-preemptive policies studied in the rest of the paper is defined.

3.2 Search space

The search space belongs to the class of Non-Preemptive "Arrival Time Dependent" policies. First, this class of policies is defined and this choice is justified.

Non-Preemptive "Arrival Time Dependent" policies The term Non-Preemptive "Arrival Time Dependent" (NP-ATD) policies comes from the fact that, within this class, the priority of a frame depends on its arrival time. NP-ATD policies is a sub-class of PPEB policies introduced earlier (see definition 1).

Definition 4 *A Non-Preemptive Arrival Time Dependent policy is a policy whose priority function can be put under the form:*

$$\Gamma_{k,n}(t) = \begin{cases} \overrightarrow{P}_{k,n} = (A_{k,n} + p_k, k, n) & \text{if } t < B_{k,n} \\ \overrightarrow{Q}_{k,n} = (-\infty, k, n) & \text{if } t \geq B_{k,n} \end{cases} \quad (1)$$

where p_k: $k \mapsto \mathbb{Q}^+$.

p_k is an arbitrary function of the stream f_k, which returns a positive value identical for all frames of f_k. The value can be an arbitrary numerical value or it can be dependent of some characteristics of the traffic stream (*i.e.* $\overline{D}_k$, T_k or C_k). For instance, for NP-EDF, p_k is equal to the relative deadline $\overline{D}_k$.

Motivations for Non-Preemptive Arrival Time Dependent policies First of all, NP-ATD policies are "good" scheduling policies:

- decidability is ensured by the last two components of the priority vectors,
- NP-ATD policies are Shift Temporal Invariant, see definition 3, since the relative priority between two frames $f_{k,n}$ and $f_{i,j}$ depends only on the offset between arrival dates and on the values returned by functions p_k and p_i,
- the policies are implementable; section 4 is devoted to the implementation issues.

Second, NP-ATD policies are promising in terms of performances. Indeed, NP-EDF, which is optimal with respect to feasibility with non-concrete traffic streams (see [34]), belongs to this class but there may exist other

policies that perform close to NP-EDF in terms of feasibility while having a much better behavior with respect to application-dependent criteria. The experiments presented in paragraph6 confirm this intuition. Third, as it will be shown in paragraph 4, a generic feasibility analysis, through response time bound computation, can be derived for all NP-ATD policies.

4 Scheduling frames with NP-ATD policies

This section deals with the practical issues of implementing NP-ATD policies for scheduling messages at the MAC level. In the following, an implementation on top of the CAN network is considered. Indeed, CAN network is widely used, very cheap and it is a well mastered technology that possesses interesting features for a wide-range of real-time systems. The mechanisms discussed in the following can be adapted in a straightforward manner to all other priority buses, such as VAN [6] and the J1850 [7].

Several studies [36, 23, 24, 25] have tackled the problem of scheduling messages with NP-EDF, which belongs to NP-ATD policies, on top of CAN. In the following, the solutions in [23] and refined later in [24, 25] is extended to NP-ATD policies.

4.1 Basics of CAN MAC protocol

On CAN, any node may start a transmission when the bus is idle. Possible conflicts are resolved by a priority-based arbitration process. In case of simultaneous transmissions, the highest priority frame will be sent despite the contention with lower priority frames. The arbitration is determined by the identifier of the frames (i.e. their priority) with the convention "the smaller the numerical value, the higher the priority". There is no need that successive instances of a recurrent frame have the same identifier. This allows the implementation of policies with dynamic priorities such as NP-EDF and NP-ATD policy.

4.2 NP-ATD policies scheduling on CAN

With NP-ATD policies, priority of frames are inversely proportional to $A_{k,n} + p_k$. The basic idea [36] is to encode the priority of the frame in the largest part of the identifier. Since it is imposed on CAN that all frames have distinct identifiers, some bits are kept to ensure the uniqueness of the identifier. A solution is to assign a bit pattern distinct for each traffic stream. However, scheduling messages with NP-ATD policies on CAN raises several problems. These problems, explicitly stated in [36], are the same as NP-EDF:

1. priorities $A_{k,n} + p_k$ become larger and larger with time since $A_{k,n}$ depends on the current clock value,
2. the mapping of the priorities to a limited number of bits induces so-called "quantization" errors (see Appendix A): two distinct priorities can be assigned the same identifier due to its limited number of bits.

Without loss of generality, the time can be considered discrete where its granularity is the bit-time (i.e., the time between the emission of two successive bits of the same frame). To solve problem 1, the authors in [36, 23, 24, 25] propose to express the priority relatively to a time origin that is increasing over time. Precisely, the time origin t_{start} is updated to each new transmission beginning (i.e. $t_{start} = \max_{k,n}(B_{k,n} \leq t)$): it is equal to the very first bit of the frame (Start of Frame bit). The priority of each pending message has to be updated at each new transmission start. This computation induces a very small overhead. For instance, it is was estimated [25] to be less than 3% on the widely used Siemens C167CR microcontroller.

An effective solution to Problem 2 was developed in [23]. The authors use a logarithmic scale to map deadlines. One can exactly reuse this technique for priorities of NP-ATD policies. The timeline is divided in so-called blocks whose size are exponentially increasing (see figure 5 in Appendix A). All blocks are subdivided into the same number of slots. The identifier of a frame is then set to the index of the slot where the deadline

falls. This scheme allows to map a set of frames having a large range of priorities to a limited number of priority bits. In [23], it is formally proven that this technique greatly improves the schedulability of the system by limiting quantization errors. Details of priority encoding using logarithmic scale applied to NP-ATD policies are given in Appendix A. Thanks to the techniques used, quantization errors is reduced but they cannot be fully avoided and thus have to be taken into account. In the following, a schedulability analysis of NP-ATD policies that includes quantization errors is proposed. It is worth pointing out that, the implementation of NP-ATD policies can be adapted to Token-Ring and CSMA-CD networks as made in [25] for NP-EDF.

5 Schedulability analysis of NP-ATD policies

Schedulability can be analysed under NP-EDF using feasibility tests [34] and computation of bounds on the Worst Case Response Times (WCRT) [37, 38]. The aim of this section is to present a schedulability analysis through WCRT bounds with arbitrary deadlines and taking into account the quantization errors. This extends the analysis made in [23, 24, 25] where only the case where deadlines are lower than periods is handled. First, paragraph 5.1 summarizes existing results on WCRT and applies them to NP-ATD policies. Then, paragraph 5.2 introduces the overhead brought by quantization errors.

5.1 WCRT under NP-EDF: a recap

The response time $r_k(\mathtt{a})$ of a frame is the time elapsed between its arrival $\mathtt{a}$ and its completion. The set of traffic streams is feasible under a given scheduling policy if the response time of each frame is lower than or equal to the relative deadline. In general, it is not possible to compute the response times of all frames for all foreseeable trajectories of the system; a solution for assessing feasibility is to compute bounds on response times. Such an analysis was derived for preemptive EDF in [33], later for NP-EDF in [37, 38]. In [37], it is shown that the worst case response time of a frame of a traffic stream occurs after a certain arrival pattern termed the "As Soon As Possible" pattern (ASAP for short). The authors use the concept of "deadline busy period" for a frame $f_{k,n}$, which is a period of network utilization without idle-time during which only frames with deadline not greater than $f_{k,n}$ are executed.

Lemma 1 *[37] A bound on the response time of a frame, belonging to traffic stream f_k, released $\mathtt{a}$ units of time after the beginning of its deadline busy period can be found in the deadline busy period induced by the ASAP pattern. The ASAP pattern is the situation where:*

- *f_k releases a frame at time $\mathtt{a}$ (other frames may have been released before time $\mathtt{a}$),*
- *all frames, which relative deadline $\overline{D}_i$ is smaller than or equal to $\mathtt{a}+\overline{D}_k$, are released from time $t=0$ on at their maximum rate,*
- *the largest frame of all traffic streams with relative deadline greater than $\mathtt{a}+\overline{D}_k$, if any, is released at time $t=-1$. This constitutes the so-called blocking factor due to non-preemptiveness.*

This lemma allows to compute a bound on the response time $r_k(\mathtt{a})$ of a traffic stream f_k released at time $\mathtt{a}$, denoted $f_k(\mathtt{a})$ in the following. It was proven [37] that the execution of $f_k(\mathtt{a})$ starts, at the latest, at the time t solution of the following equation which can be solved by recurrence:

$$t = W_k(\mathtt{a}, t) + \mathcal{B}_k(\mathtt{a}) + \left\lfloor \frac{\mathtt{a}}{T_k} \right\rfloor \cdot C_k. \tag{2}$$

in which t is the length $L_k(\mathtt{a})$ of the "deadline busy period", and where $W_k(\mathtt{a}, t)$ is an upper bound for the "higher priority workload" (i.e. work induced by frames of lower or equal deadlines) in an interval of length t

$((x)^+ \stackrel{\text{def}}{=} \min(x,0))$:

$$W_k(\mathtt{a},t) = \sum_{i \neq k} \underbrace{\left(\min\left(\left\lfloor \frac{t}{T_i} \right\rfloor + 1, \left\lfloor \frac{\mathtt{a} + \overline{D_k} - \overline{D_i}}{T_i} \right\rfloor + 1\right)\right)^+}_{\substack{\text{maximum number of instances in interval } t \\ \text{with a deadline lower than or equal to } \mathtt{a} + \overline{D}_k}} .C_i \tag{3}$$

with $\mathcal{B}_k(\mathtt{a})$ the blocking factor (i.e. the longest frame having a greater deadline than frame $f_k(\mathtt{a})$):

$$\mathcal{B}_k(\mathtt{a}) = \max_{i=1..m} \{C_i \mid \overline{D}_i > \mathtt{a} + \overline{D}_k\} - 1 \tag{4}$$

and where the work of the other frames of traffic stream f_k is:

$$\left\lfloor \frac{\mathtt{a}}{T_k} \right\rfloor \cdot C_k \tag{5}$$

Thus, a bound on the response time of a frame $f_k(\mathtt{a})$ is:

$$r_k(\mathtt{a}) = \max\{C_k, L_k(\mathtt{a}) + C_k - \mathtt{a}\}. \tag{6}$$

The response time bound for f_k is $\max_{\mathtt{a}} r_k(\mathtt{a})$. It would be very time consuming to compute $r_k(\mathtt{a})$ for all values of $\mathtt{a}$ but it is proven in [33, 37] that the only "significant" values of $\mathtt{a}$ are the elements of the following set $\mathbf{A}_k$:

$$\mathbf{A}_k = \{t = n \cdot T_i + \overline{D}_i - \overline{D}_k \mid t \geq 0,\, t \leq \mathcal{L} - C_k,\, n \in \mathbb{N},\, i = 1...m\} \tag{7}$$

where $\mathcal{L}$ is the longest busy period (longest duration of the resource without idle time, see [33] for computation details). The significant values are the values which imply changes in the higher priority workload computed with equation 3. The next section shows how this analysis can be adapted when priority inversions may occur due to the limited number of bits available for encoding the deadlines.

5.2 WCRT under NP-EDF with quantization error

When the logarithmic scale is used, frames which have distinct deadlines can belong to the same slot. The initial priority ordering given by deadlines is then lost and priority inversions may take place. In the worst-case, when computing the WCRT bound of a frame $f_k(\mathtt{a})$, all frames having a deadline in the same slot are assumed to be of higher priority than $f_k(\mathtt{a})$. Precisely, the higher priority workload is now made of all frames having a deadline lower or equal than $A_{k,n} + \overline{D}_k + S_k(\mathtt{a},t)$, where $S_k(\mathtt{a},t)$ is a bound on the size of the slot where $\mathtt{a} + \overline{D}_k - t_{start}$ falls. Thus the higher priority workload $W_k(\mathtt{a},t)$ becomes:

$$W_k(\mathtt{a},t) = \sum_{i \neq k} \left(\min\left(\left\lfloor \frac{t}{T_i} \right\rfloor + 1, \left\lfloor \frac{\mathtt{a} + \overline{D}_k + S_k(\mathtt{a},t) - \overline{D}_i}{T_i} \right\rfloor + 1\right)\right)^+ \cdot C_i \tag{8}$$

A justification of the above formula is given in Appendix B. A difficulty is that the length of slot $S_k(\mathtt{a},t)$ is not constant over time: it depends on the value of the deadline that is to be encoded (i.e. $\mathtt{a} + \overline{D}_k$), and on the time origin of the logarithmic scale (t_{start}), which is updated at each new transmission beginning. As explained in Appendix B, the size of $S_k(\mathtt{a},t)$ increases monotonically with $\mathtt{a} + \overline{D}_k - t_{start}$. To derive an upper bound on $S_k(\mathtt{a},t)$, one has to find a lower bound on the value of t_{start} at which $f_k(\mathtt{a})$ may gain the channel. Since $f_k(\mathtt{a})$ arrives in $\mathtt{a}$, an obvious lower bound is $\mathtt{a}$ but one can improve that result according to the scheme proposed in [24].

Which a to analyze As shown in [33, 37], the only significant values of a that are really needed to be analyzed are values which imply changes in the higher priority workload computed by equation 8. Precisely, these values are integral values of the form $n \cdot T_i + \overline{D}_i - \overline{D}_k - S_k(\mathtt{a}, t)$; a problem is that the length of the slot $S_k(\mathtt{a}, t)$ is a priori unknown since it depends also on $\mathtt{a}$. The logarithmic encoding scheme presented in Appendix A ensures that, at any time, the length of a slot takes its value in $[U, 2^{k_{max}}U]$, where U is the length of the first time slot. A solution to capture a change in the higher priority workload is thus to consider all possible values for the slot's length; the set of $\mathtt{a}$ to analyse becomes:

$$\mathbf{A}_k = \{t = n \cdot T_i + \overline{D}_i - \overline{D}_k - x \mid t \geq 0,\, t \leq \mathcal{L} - C_k,\, n \in \mathbb{N},\, x \in \mathbb{N},\, x = U...2^{k_{max}}U,\, i = 1...m\}. \quad (9)$$

The next section shows how this analysis can be easily adapted to Arrival Time Dependent policies.

5.3 WCRT for NP-ATD policies with quantization error

A frame $f_{k,n}$ under NP-EDF possesses an initial priority vector $\overrightarrow{P}_{k,n,}$ equal to $(A_{k,n}+\overline{D}_k, k, n)$; NP-EDF is thus a particular case of NP-ATD policy where $p_k : k \mapsto \overline{D}_k$ (see definition of NP-ATD policies in paragraph 3.2). In the following, it will be shown that Lemma 1 as well as the set of arrival dates to consider after the ASAP pattern (see equation 7) remain valid with the condition that $\overline{D}_k$ is replaced by p_k. "Deadline busy periods" becomes "priority busy periods" for a frame $f_{k,n}$ which are intervals of network utilization without idle-time during which only instances with higher priority than $f_{k,n}$ are executed.

Lemma 2 *A bound on the response time of a frame, belonging to traffic stream f_k, released $\mathtt{a}$ units of time after the beginning of its priority busy period can be found in the priority busy period induced by the ASAP pattern. The ASAP pattern is the situation where:*

- *f_k releases a frame at time $\mathtt{a}$ (other frames may have been released before time $\mathtt{a}$),*
- *all frames, for which p_i is smaller than or equal to $\mathtt{a} + p_k$, are released from time $t = 0$ on at their maximum rate,*
- *the largest frame of all traffic streams with p_i greater than $\mathtt{a} + p_k$, if any, is released at time $t = -1$. This constitutes the so-called blocking factor due to non-preemptiveness.*

Sketch Of Proof:

Consider virtual NP-EDF, a modified version of NP-EDF that would schedule frames not by taking into account their deadline $A_{k,n} + \overline{D}_k$ but an arbitrary "virtual" deadline $A_{k,n} + p_k$. The priority function of this virtual NP-EDF would be:

$$\Gamma_{k,n}^{virtual\,NP-EDF}(t) = \begin{cases} \overrightarrow{P}_{k,n} = (A_{k,n} + p_k,\, k,\, n) \text{ if } t < B_{k,n} \\ \overrightarrow{Q}_{k,n} = (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases},$$

where p_k, as $\overline{D}_k$, possesses the property that its value is equal for all frames of traffic stream f_k. Indeed, this property on p_k is needed for lemma 4.1 in [33] to hold (precisely, when building the ASAP pattern, shifting left a frame must increase the higher priority workload).

According to lemma 1, a response time bound for f_k under virtual-NP-EDF occurs after the ASAP pattern, as defined by Spuri [33], where $\overline{D}_k$ is replaced by p_k in the equations 3 and 7. To assess the feasibility, the response time bounds just have to be compared with the actual relative deadlines $\overline{D}_k$.

■

The way to compute the worst case response times under a NP-ATD policy is the same as under NP-EDF except that $\overline{D}_k$ is replaced by the p_k value corresponding to the policy in equations 4 and 8.

6 Experiments

In this following, the performances of NP-ATD scheduling policies are assessed in the context of Networked Control Systems, which is an important application field of our proposal.

6.1 Search space

The aim is to find scheduling policies that perform well in terms of feasibility, for optimizing the use of the network bandwidth, but that are also efficient with respect to the criteria important for NCS (defined in 6.2.1 and 6.3.2). In the following, simulations are done within a sub-class of NP-ATD policies having a priority vector $\overrightarrow{P}_{k,n}$ expressed as:

$$\overrightarrow{P}_{k,n} = (A_{k,n} + c \cdot C_k + d \cdot \overline{D_k},\ k,\ n) \text{ with } c \in [0, 50] \text{ and } d \in [0, 1], \tag{10}$$

which means that, in definition 4, one has $p_k : k \rightarrow c \cdot C_k + d \cdot \overline{D_k}$. Thus, a policy belonging to our search space is defined by a priority function having the form of equation 10. Simulations have shown that parameters $c \in [0, 50]$ and $d \in [0, 1]$ define a search space where, most generally, the policies of interest are to be found.

This sub-class of NP-ATD policies was chosen because it is expected to contain policies providing a good trade-off between feasibility and the satisfaction of the other criteria important for NCS (see 6.2.1). Indeed, NP-EDF actually belongs to this class ($p_k = \overline{D}_k$) and policies whose priority function is "close" to NP-EDF are expected to perform similarly in terms of schedulability. On the other hand, introducing a term that depends on the transmission time should help to improve the other criteria. In particular, it has been shown that preemptive Shortest Remaining Processing Time First (SRPTF) is optimal for average response times (see [39, 40] quoted in [41]) and, in our simulations, Non-Preemptive Shortest Maximum Processing Time First (NP-SMPTF), which is the non-preemptive version of SRPTF defined with $\overrightarrow{P}_{k,n} = (C_k,\ k,\ n)$, outperformed NP-EDF for all considered criteria besides feasibility.

In the following, the performances of NP-ATD policies are evaluated against NP-SMPTF and NP-EDF. In addition, Non-Preemptive Deadline Monotonic (NP-DM) is also considered because it is known for its good performances in terms of schedulability[1] and its ease of implementation.

6.2 Performance wrt scheduling

6.2.1 Performance criteria

Classically, a control loop is comprised of the sampling (i.e. data are read from sensors), the computation of the control, and the actuation (i.e. transmission of the control outputs to the actuators). Specific delays have been identified to impact the stability and, more generally, the performances of the system (see, for instance, studies in [42, 1, 3]). These delays include:

- *the input-output latency*: the time elapsed between the sampling and the actuation,
- *the sampling interval*: the time interval between two consecutive sampling points,
- *the sampling latency:* the time elapsed between the theoretical sampling time and its actual occurrence.

In NCS, where the control loop is distributed over a network, data transfered from the sensors to the controller, and from the controller to the actuators are exchanged over a network. The communications from sensors to controller (response time denoted by t_{sc}) and from controller to actuators (response time denoted by t_{ca}) induce non-constant delays in the input-output latency (equal to t_{sc}+computation time+t_{ca}), and its variability is known to have a crucial influence on the control-loop performances (see, for instance, [1] and experiments

[1] Non-Preemptive Deadline Monotonic is optimal wrt to feasibility for the non preemptive scheduling with $\overline{D}_k \leq T_k$ when $\overline{D}_k \leq \overline{D}_i$ implies $C_k \leq C_i$, see [37].

of paragraph 6.3). An efficient communication systems should thus minimize communication delays and their variabilities. In the experiments that follow, the impact of NP-ATD policies on the response times of the traffic streams (i.e., delay between the queuing time and the transmission end) is studied.

6.2.2 Simulation setup

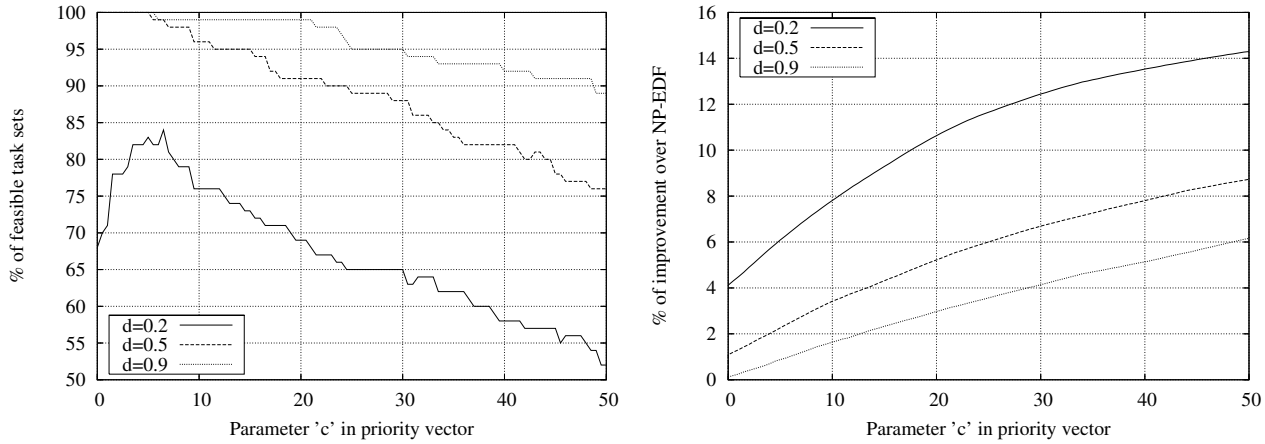
One considers several traffic streams exchanged on a CAN network. As to our best knowledge there are no analytic techniques for evaluating the values of our criteria, simulation is employed. A given criterion is evaluated for a policy as the average value over all traffic streams.

The byte transmission time is the smallest time unit, which means that the propagation delay and the bit-stuffing are neglected (since bit-stuffing depends on the actual data transmitted, it is out of the scope of this study). The transmission time of a frame, denoted by C_k, is randomly chosen between 8 and 16 according to an uniform law. The utilisation rate ($\frac{C_k}{T_k}$) of traffic stream f_k is uniformally distributed in $\left[\frac{U}{m} \cdot 0.9\ ,\ \frac{U}{m} \cdot 1.1\right]$ where U is the network load and m the total number of traffic streams. The relative deadlines $\overline{D}_k$ are uniformally chosen in the interval $\left[T_k - \frac{T_k - C_k}{2}, T_k + \frac{T_k - C_k}{2}\right]$. Simulation consists in scheduling concrete sets of traffic streams, which are generated from non-concrete ones by randomly choosing the initial offset for each traffic stream f_k in the interval $[0, T_k]$. Response time bound computations and simulations have been implemented in C++, and an applet version of the simulator is freely available at *http://www.loria.fr/~grenier/logiciel/SimApplet.html*.

6.2.3 Influence of parameters

In this paragraph it is assessed how the parameters c and d influence feasibility and average response time jitters, as measured by the standard deviation of the response times. For the former criterion, the performance for a policy $\mathcal{A}$ is the percentage of traffic streams that are feasible under NP-EDF and that remains feasible under $\mathcal{A}$.

Figures 1(a) and 1(b) shows the performances of the set of policies defined by equation 10 where $d \in \{0.2, 0.5, 0.9\}$ and c takes its value in $[0, 50]$ with steps equal to 0.5 . The performances are presented in terms of feasibility in figure 1(a) and average response time jitter in figure 1(b). The average values are computed over 1500 simulation runs: 100 non-concrete sets of 10 traffic streams with 15 different offsets with a global load randomly chosen in the interval $[0.6, 0.7]$. Only task sets that are feasible under NP-EDF were selected.



(a) Feasibility

(b) Average response time jitters

Figure 1: Feasibility and average response time jitters for policies defined by equation 10 with $d \in \{0.2, 0.5, 0.9\}$ and c ranging from 0 to 50. Each point is the average value over 1500 simulations of task sets of cardinality 10 with an average load of 0.65.

Two guidelines can be drawn from figure 1:

- the larger the value of c in $\overrightarrow{P}_{k,n}$, the better the average response time jitters. The counterpart is that feasibility significantly diminishes when c increases.

- the larger the value of d in $\overrightarrow{P}_{k,n}$, the better the feasibility with the drawback that reponse time jitters increase.

Indeed, when c becomes large in equation 10, $c \cdot C_k$ has more influence than $A_{k,n}$ and $d \cdot \overline{D}_k$ on the frame priority. Thus, the policy tends to behave in a similar manner as Non-Preemptive Shortest Maximum Processing Time First (see paragraph 6.1). The same reason explains that when d increases, the policy performs close to NP-EDF for both criteria. It is worth noting that no significant differences among policies for the average response times are observed. This fact is due to the relatively small variations of transmission time C_k imposed by the CAN protocol.

For the sake of comparison, table 1 presents the performances of NP-SMPTF and NP-DM by comparison to NP-EDF with the same simulated task sets. From figure 1 and table 1, one sees that NP-ATD policies clearly

	Feasibility (%)	Avg response time jitter : improvement over NP-EDF
NP-SMPTF	57.5	17.9
NP-DM	100	−4.5

Table 1: Feasibility and average response time jitters (in %) for NP-SMPTF and NP-DM.

outperforms NP-SMPTF in terms of feasibility (when c lower than 40), and is better than NP-DM and NP-EDF for the response time jitters. For instance, the policy defined by $\overrightarrow{P}_{k,n} = (A_{k,n} + 18 \cdot C_k + \frac{2}{10}\overline{D}_k, k, n)$ leads to feasible schedules for about 70% of the sets of traffic streams while achieving an average improvement of 10.2% over NP-EDF for the response time jitter. On the same simulation setup, NP-SMPTF is feasible for 57.5% of the traffic stream sets while improving jitters of 17.9%. On the other hand, NP-DM leads to 100% feasible schedules but is worst than NP-EDF regarding the jitters.

In practice, most NCS will have better performances (i.e., lower response time jitter in the simulations) with a well-chosen NP-ATD policy ensuring feasibility than under NP-EDF or NP-DM. Furthermore, feasibility with NP-ATD policy is much better than under NP-SMPTF. NP-ATD policies enable the application designer to implement a MAC level protocol providing a good trade-off between feasibility and application-dependent criteria such as jitter minimization.

6.3 Control performance

In this part, a particular NCS is studied and the performances of the best feasible NP-ATD policy found are compared to the performances of NP-EDF. The process is a DC servo with a transfer function defined as $G(s) = \frac{1000}{s^2+s}$. The system, see figure 2, consists of a control-loop where the sensor, the controller and the actuator nodes are distributed over a CAN network at 125kbit/s. There is also an interfering node generating network traffic that models the other real-time traffic streams exchanged on the bus. This system is simulated with the toolbox TrueTime under Matlab/Simulink, and is described in more detail in [43]. For this study, the NP-ATD MAC layer protocol described in section 4.1 has been implemented in TrueTime.

6.3.1 Architecture overview

Precisely, as in [43], the NCS is constituted of:

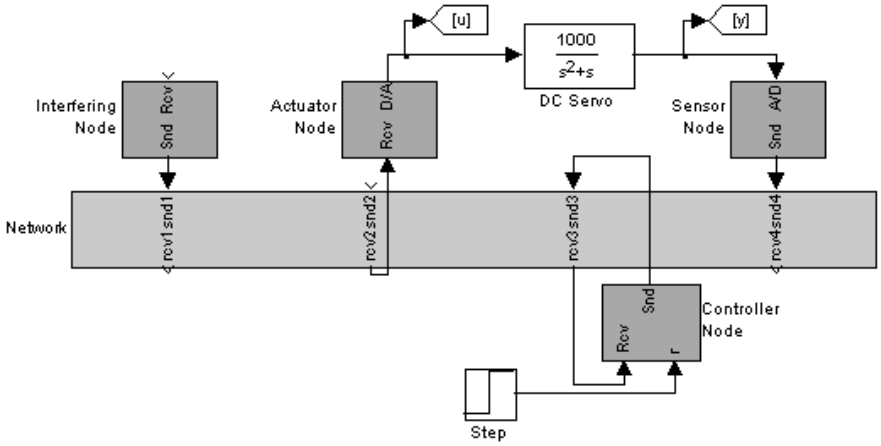


Figure 2: The architecture of the NCS consists of a control-loop where the sensor, the controller and the actuator nodes are distributed over a CAN network at 125kbit/s. An interfering node generates disturbing traffic.

- a sensor node (one task - periodic) where the sensor task samples the process with a period of 0.01s. Then, the task sends the result $y(k)$ to the controller over the network ($C = 80$bits). The delay between the sampling time and the queuing time is set to $9 \cdot 10^{-4}$s.

- a controller node (one task -event driven) where the controller task, each time it receives a frame from the sensor, computes the control output signal $u(k)$ and sends it to the actuator. The delay between the reception of the frame and the queuing of the control signal frame ($C = 80$bits) is set to $5 \cdot 10^{-4}$s. A PD-controller is used, and implemented according to the following equations:

$$u(k) = P(k) + D(k),$$

$$P(k) = 1.5 \cdot (r(k) - y(k)),$$

$$D(k) = 3.5 \cdot 10^{-5} \cdot D(k-1) + 5.25 \cdot (y(k-1) - y(k)),$$

  where $r(k)$, the reference, is a unit step (see [43] and [44] for more details).

- an actuator node (one task - event-driven) controls the actuator according to the data received in the controller's frame. The delay between the reception of the frame and the actuation is $5 \cdot 10^{-4}$s.

- the interfering node (one task - time-driven) generates the disturbing traffic. The set of its traffic streams is randomly generated as follows: for a given load U, a random traffic stream is iteratively added until the load of the node becomes higher than U. The period of each traffic stream is set at random according to the uniform law in the set $\{0.01, 0.02, 0.05, 0.1, 0.5\}$, and the number of data bytes takes a random value in the interval $[1, 8]$.

6.3.2 Performance criteria

The unit step response of the system (in our simulation, if $t < 0$, $r(t) = 0$ else $r(t) = 1$) is used with the overshoot $\overline{O}$, the settling time $T_{settling}$ and the IAE (Integral of the Absolute magnitude Error) as the performance metrics. In this study, $IAE = \int_0^{T_{sim}} |e(t)|\, dt$, where $T_{sim} = 0.5$s is the duration of the simulation, and the settling time $T_{settling}$ is the time required for the system to settle within 10%.

6.3.3 Simulation results

Figure 3 shows an instance of the unit step response of the system (at a load $U = 0.5$) with the best feasible NP-ATD policy of the search space and with NP-EDF. As one can see , the overshoot is reduced from 1.155 to 1.085, while $T_{settling}$ drops from $46 \cdot 10^{-3}$ to $119 \cdot 10^{-3}$ with the best feasible NP-ATD policy.

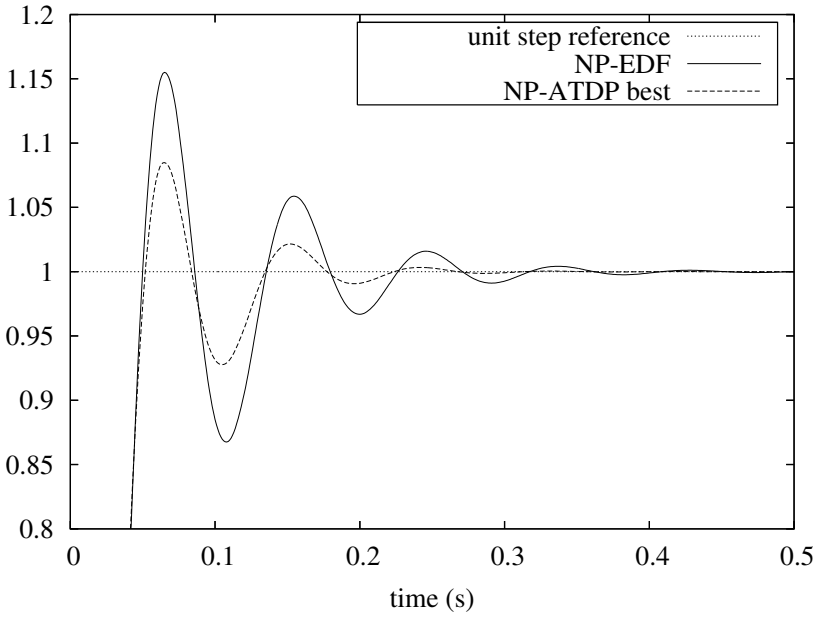


Figure 3: Unit step response of the NCS with NP-EDF and the best feasible NP-ATD policy found by an exhaustive search ($c = 35$ and $d = 0.4$). The load is equal to 0.5 .

Figure 4 presents the average performances improvement (over 100 runs for each point) of the best feasible NP-ATD policy found over NP-EDF for the three performance metrics of interest. It provides clear-cut evidences of the improvement brought by NP-ATD over NP-EDF for the NCS under study. Indeed, the improvement is significant whatever the load and increases with it. For instance, at a load equal to 0.6, the improvement over NP-EDF is 35.5% for the settling time, 30.1% for the overshoot and 14.3% for the IAE.

7 Conclusion and future work

This paper introduces the class of Non-Preemptive Arrival Time Dependent (NP-ATD) policies intended for the scheduling of frames at the MAC level. These policies are easily implementable on COTS components (e.g., CAN controllers) and provide a good trade-off between feasibility and the satisfaction of other application-dependent criteria such as the response time jitter. An NP-ATD policy can be "built" for the performance requirements of a particular application or be chosen to provide good average performances on random sets of traffic streams.

In order to assess the feasibility of systems using NP-ATD policies, a schedulability analysis that is generic for all policies of the class was proposed. In the context of Networked Controlled Systems where delays and jitters impact the performances of the control loop, simulations have shown that well chosen policies can bring significant improvements over plain NP-EDF or NP-DM.

In a future work, a more accurate evaluation of the impact of the scheduling policies on the controlled system is intended to be done on real platforms. Another improvement would be to come up with more efficient

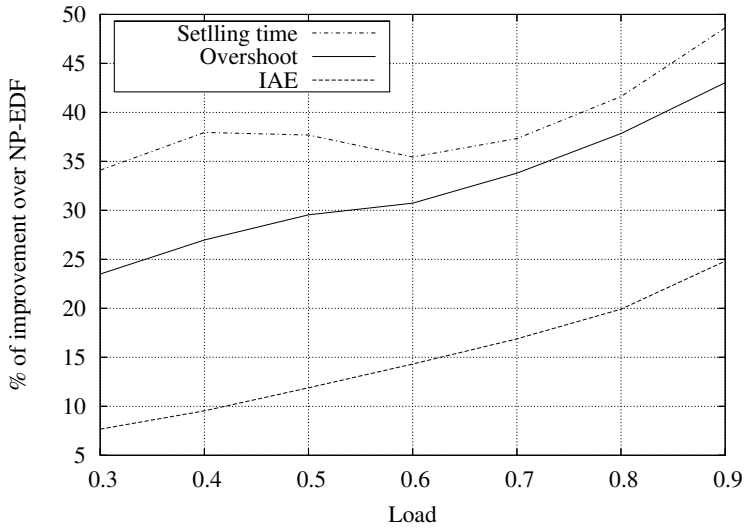


Figure 4: Average improvement of overshoot, settling time and IAE over NP-EDF with the best feasible NP-ATD policy found by an exhaustive search.

search techniques for exploring the policy search space; preliminary experiments have shown that a simple neighbourhood technique such a hill-climbing with a Tabu list is more efficient than the exhaustive search. Finally, this work could be extended to other class of policies such as time-sharing policy (e.g. Round-Robin [45], Pfair [46]); the main problem will be here to come up with a schedulability analysis valid for all policies of the class.

References

[1] B. Wittenmark, J. Nilsson, and M. Törngren, "Timing problems in real-time control systems," in *Proc. of the American Control Conference*, vol. 3, pp. 2000–2004, 1995.

[2] M. Törngren, "Fundamentals of implementing real-time control applications in distributed computer systems," *Real-Time Systems*, vol. 14, no. 3, pp. 219–250, 1998.

[3] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Proc. of the 39th IEEE Conference on Decision and Control*, (Sydney, Australia), 2000.

[4] R. Bosch, "CAN Specification Version 2.0," 1991.

[5] ISO, "ISO International Standard 11898 - Road vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed Communications," 1993.

[6] ISO, "ISO International Standard 11519-3 - Road vehicles - Low-speed serial data communication - Vehicle Area Network (VAN)," 1994.

[7] SAE, "SAE J1850 Class B data communication network interface," 1992.

[8] A. Cervin and J. Eker, "Control-scheduling codesign of real-time systems: The control server approach," *Journal of Embedded Computing*, vol. 1, no. 2, 2004.

[9] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proc. of the 19th IEEE Real-Time Systems Symposium (RTSS 1998)*, 1998.

[10] G. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Proc. of the 19th IEEE Real-time Systems Symposium (RTSS 1998)*, pp. 286–295, 1998.

[11] M. Caccamo, G. Buttazzo, and L. Sha, "Elastic feedback control," in *Proc. of the 12th Euromicro Conference on Real-Time Systems (Euromicro-RTS 2000)*, pp. 121–128, 2000.

[12] A. Crespo, I. Ripoll, and P. Albertos, "Reducing delays in RT control: the control action interval," in *Proc. of the 14th IFAC World Congress*, 1999.

[13] A. Cervin, *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, 2003.

[14] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari, "Scheduling periodic task systems to minimize output jitter," in *Proc. of the 6th International Conference on Real-Time Computing Systems and Applications (RTCSA 1998)*, (Hong Kong, China), pp. 62–69, 1998.

[15] L. David, F. Cottet, and N. Nissanke, "Jitter control in on-line scheduling of dependent real-time tasks," in *Proc. of the 22nd IEEE Real-time Systems Symposium (RTSS 2001)*, pp. 49–58, 2001.

[16] J. Goossens and P. Richard, "Performance optimization for hard real-time fixed priority tasks," in *Proc. of the 12th International Conference on Real-Time Systems (RTS 2004)*, pp. 241–258, 2004.

[17] N. Navet and J. Migge, "Fine tuning the scheduling of tasks through a genetic algorithm: Application to Posix1003.1b compliant OS," *IEE Proceedings Software*, vol. 150, no. 1, pp. 13–24, 2003.

[18] K. Altisen, G. Goessler, A. Pnueli, J. Sifakis, S. Tripakis, and S. Yovine, "A framework for scheduler synthesis," in *Proc. of the 20th IEEE Real-Time Systems Symphosium (RTSS 1999)*, pp. 154–163, 1999.

[19] E. Grolleau and A. Choquet-Geniet, "Scheduling real-time systems by means of petri nets," in *Proc. of the IFAC 25th Workshop on Real-Time Programming (WRTP'00)*, (Palma de Mallorca), pp. 95–100, 2000.

[20] M. Grenier and N. Navet, "New on-line preemptive scheduling policies for improving real-time behavior," in *Proc. of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005)*, vol. 2, (Catania, Italy), pp. 315–322, sep 2005. Available at `http://www.loria.fr/~nnavet/publi/etfa2005_mgnn.pdf`.

[21] T. Nolte, M. Nolin, and H. Hansson, "Real-time server-based communication for CAN," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 3, pp. 192–201, 2005.

[22] G. Cena and A. Valenzano, "Achieving round-robin access in controller area networks," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 6, pp. 1202–1213, 2002.

[23] A. Meschi, M. D. Natale, and M. Spuri, "Earliest Deadline message scheduling with limited priority inversion," in *Proc. of the 4th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS '96)*, pp. 87–94, 1996.

[24] M. D. Natale, "Scheduling the CAN bus with Earliest Deadline techniques," in *Proc. of the 21st IEEE Real-time Systems Symposium (RTSS 2000)*, pp. 259–268, 2000.

[25] M. D. Natale and A. Meschi, "Scheduling messages with Earliest Deadline techniques," *Real-Time Systems*, vol. 20, no. 3, pp. 255–285, 2001.

[26] P. Pedreiras and L. Almeida, "Flexible scheduling on controller area network," in *Proc. of the 10th International Conference on Real-Time Systems (RTS 2002)*, (Paris, France), 2002.

[27] S. Cavalieri, "Meeting real-time constraints in CAN," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 124–135, 2005.

[28] CiA, "CAN application layer for industrial applications: CAN in the OSI reference model," 1996.

[29] CiA, "CAL-Based communication profile for industrial systems-CANOpen." Version 3, 1996.

[30] T. Nolte, H. Hansson, and C. Norström, "Minimizing CAN response-time jitter by message manipulation," in *Proc. of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, (San Jose, California, USA), pp. 197–206, IEEE Computer Society, September 2002.

[31] F. Coutinho, J. Fonseca, J. Barreiros, and E. Costa, "Using genetic algorithms to reduce jitter in control variables transmitted over CAN," in *Proc. of the 7th International CAN Conference (ICC 2000)*, (Amsterdam, The Netherlands), oct 2000.

[32] J. Barreiros, E. Costa, J. Fonseca, and F. Coutinho, "Jitter reduction in a real-time message transmission system using genetic algorithms," in *Proc. of Congress on Evolutionary Computation (CEC'2000)*, (La Jolla Marriott, San Diego, USA), 2000.

[33] M. Spuri, "Analysis of deadline scheduling in real-time systems," Tech. Rep. RR-2772, INRIA, 1996. Available at `http://www.inria.fr/rrrt/rr-2772.html`.

[34] K. Jeffay, D. Stanat, and C. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," in *Proc. of the 12th IEEE Real-time Systems Symposium (RTSS 1991)*, pp. 129–139, 1991.

[35] J. Migge, *Scheduling under Real-Time Constraints: a Trajectory Based Model.* PhD thesis, University of Nice Sophia-Antipolis, 1999.

[36] K. Zuberi and K. Shin, "Non-preemptive scheduling of messages on Controller Area Network for real-time control applications," in *Proc. of the Real-Time Technology and Applications Symposium (RTAS'95)*, pp. 240–249, 1995.

[37] L. George, N. Rivierre, and M. Spuri, "Preemptive and non-preemptive real-time uniprocessor scheduling," Tech. Rep. RR-2966, INRIA, 1996. Available at `http://www.inria.fr/rrrt/rr-2966.html`.

[38] J.A.Stankovic, M.Spuri, K. Ramamritham, and G.C.Buttazzo, *Deadline Scheduling for Real-Time Systems EDF and Related Algorithms.* Kluwer Academic Publishers ISBN 0-7923-8269-2, 1998.

[39] L. Schrage, "A proof of the optimality of the shortest remaining time discipline," *Operations Research*, vol. 16, pp. 687–690, 1968.

[40] D. Smith, "A new proof of the optimality of the shortest remaining time discipline," *Operations Research*, vol. 26, no. 1, pp. 197–199, 1976.

[41] D. P. Bunde, "SPT is optimally competitive for uniprocessor flow," *Information Processing Letters*, vol. 90, no. 5, pp. 233–238, 2004.

[42] K. Astrom and B. Wittenmark, *Computer-Controlled Systems.* Prentice Hall ISBN 0-13-168600-3, third ed., 1997.

[43] M. Ohlin, D. Henriksson, and A. Cervin, *TrueTime 1.5 Reference Manual.* Department of Automatic Control, Lund University, Sweden, Jan 2007.

[44] Åström, K. Johan, and T. Hägglund, *Advanced PID Control.* Research Triangle Park, North Carolina: ISA - The Instrumentation, Systems, and Automation Society, 2005.

[45] J. Migge, A. Jean-Marie, and N. Navet, "Timing analysis of compound scheduling policies : Application to Posix1003.1b," *Journal of Scheduling*, vol. 6, no. 5, pp. 457–482, 2003.

[46] J. Anderson and A. Srinivasan, "Pfair scheduling: Beyond periodic task systems," in *Proc. of the 7th International Conference on Real-time Computing Systems and Applications*, pp. 297–306, IEEE Computer Society Press, dec 2000.

A Logarithmic scale to map priority

This appendix presents the logarithmic scale, introduced in [23] for the deadline case, to map the priority $A_{k,n} + p_k$ of a frame $f_{k,n}$ to a limited number n of priority bits in order to reduce quantization errors by providing finer resolution for closer deadline.

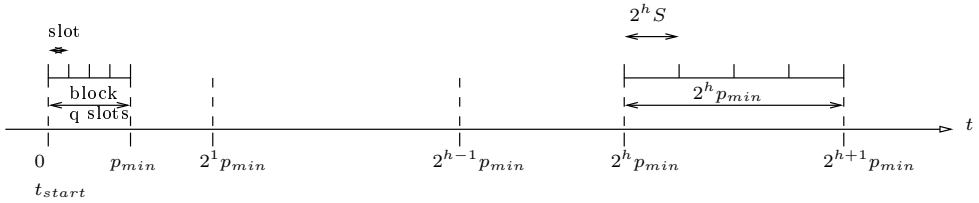


Figure 5: Logarithmic scale starting at t_{start}, where p_{min} is the minimum relative priority in the traffic stream, h is the index of the block computed with equation 11 and S is the size of the first time slot (*i.e.* $\frac{p_{min}}{q}$).

An example of a logarithmic scale is shown in figure 5. The time origin of the logarithmic scale, denoted by t_{start}, is updated at each new transmission beginning ($t_{start} \stackrel{\text{def}}{=} \max_{k,n}(B_{k,n} \leq t)$, where t is the current time). Timeline, from t_{start} to p_{max} ($p_{max} \stackrel{\text{def}}{=} \max_{\tau_k \in \mathcal{T}}(p_k)$) is divided in blocks of increasing length. Each block is divided into q slots where q is an integer known at the design stage (see equation 12). The priority of a frame corresponds to the index of the slot containing the current deadline $A_{k,n} + p_k - t_{start}$. Index i, from [23], is computed as follow:

$$i = h \cdot q + \left\lfloor \frac{A_{k,n} + p_k - t_{start} - 2^h p_{min}}{2^h S} \right\rfloor,$$

where S is the size of the first time slot ($S \stackrel{\text{def}}{=} \frac{p_{min}}{q}$ and $p_{min} \stackrel{\text{def}}{=} \min_{\tau_k \in \mathcal{T}}(p_k)$). The index of the block, called h, where the current deadline $A_{k,n} + p_k - t_{start}$ falls, from [23], is:

$$h = \begin{cases} \left\lfloor log_2 \frac{A_{k,n}+p_k-t_{start}}{p_{min}} \right\rfloor & \text{if } A_{k,n} + p_k - t_{start} > p_{min} \\ 0 & \text{otherwise} \end{cases}. \quad (11)$$

The number of slots q in a block (see Appendix A of [23] for computation details) is given by:

$$q = \left\lfloor \frac{2^n}{\lfloor log_2 k_{max} \rfloor + \frac{k_{max}}{2^{\lfloor log_2 k_{max} \rfloor}}} \right\rfloor, \quad (12)$$

where $k_{max} \stackrel{\text{def}}{=} \frac{p_{max}}{p_{min}}$. By definition, the size of the slot where $A_{k,n} + p_k - t_{start}$ "falls" is equal to $2^h S$.

When the logarithmic scale is used (or when a limited number of bits are devoted for priority encoding), frames which have distinct deadlines can belong to the same slot and an inversion of priority may occur. In such a case, these errors are called quantization errors.

B Higher priority workload $W_k(\mathtt{a}, t)$

This appendix details computation of higher priority workload $W_k(\mathtt{a}, t)$ when priority inversions due to the limited number of bits available for encoding the deadlines occur.

The higher priority workload $W_k(\mathtt{a},t)$ is, by definition, the work induced by frames of traffic stream f_i, different than f_k, in a deadline busy period of length t. A deadline busy period is a period of network utilization without idle-time during which only frames with priority greater than $f_k(\mathtt{a})$ are executed. In the following, u is the beginning of the deadline busy period. $A_{i,j_0} = \min\{A_{i,j} \mid j \in \mathbb{N}, A_{i,j} \geq u\}$ and $A_{i,j_1} = \max\{A_{i,j} \mid j \in \mathbb{N},\ A_{i,j} \leq u+t\}$ are respectively the first and the last frame of the traffic stream f_i in the deadline busy period. The number of frames of f_i in the interval $[u, u+t]$ is $n = j_1 - j_0 + 1$.

The goal is to bound, for each traffic stream f_i, the work induced by frames of f_i in $[u, u+t]$. The work of the last frame of traffic stream f_i arrived at $A_{i,j_1}$is take into account iff:

1. f_{i,j_1}arrives before the end of the interval $[u, u+t]$: $A_{i,j_1} \leq u+t$.

$$A_{i,j1} \leq u+t$$
$$A_{i,j_0} + (n-1)\cdot T_i \leq u+t$$
$$\text{Thus: } n = \left\lfloor \frac{t}{T_i} \right\rfloor + 1 \leq \frac{t}{T_i} + 1 \text{ because } n \in \mathbb{N} \text{ and } A_{i,j_0} \geq u$$

2. Furthermore, the last frame f_{i,j_1} must have a priority greater than $f_k(\mathtt{a})$. For this purpose, we assume that all frames having a deadline belonging to the same slot as $A_{k,n} + \overline{D}_k - t_{start}$ are of higher priority than $f_k(\mathtt{a})$. Thus, frames having a deadline lower or equal than $A_{k,n} + \overline{D}_k + S_k(\mathtt{a},t)$ are considered to have a higher priority ($S_k(\mathtt{a},t)$ is an upper bound of the size of the slot to which $A_{k,n} + \overline{D}_k - t_{start}$ belongs, see [24] for computation details).

$$A_{i,j_1} + \overline{D}_i - t_{start} \leq u + \mathtt{a} + \overline{D}_k + S_k(\mathtt{a},t) - t_{start}$$
$$\text{Thus: } n = \left\lfloor \frac{\mathtt{a} + \overline{D}_k + S_k(\mathtt{a},t) - \overline{D}_i}{T_i} \right\rfloor + 1 \leq \frac{\mathtt{a} + \overline{D}_k + S_k(\mathtt{a},t) - \overline{D}_i}{T_i} + 1 \text{ because } n \in \mathbb{N}$$

Hence, an upper bound of the number of frames n induced by a traffic stream f_i in a deadline busy period of length t is: $\min\left(\left\lfloor \frac{t}{T_i} \right\rfloor + 1, \left\lfloor \frac{\mathtt{a}+\overline{D}_k+S_k(\mathtt{a},t)-\overline{D}_i}{T_i} \right\rfloor + 1\right)$. A bound on the higher priority work $W_k(\mathtt{a},t)$, due to traffic streams different from f_k, is thus:

$$W_k(\mathtt{a},t) = \sum_{i \neq k} \left(\min\left(\left\lfloor \frac{t}{T_i} \right\rfloor + 1, \left\lfloor \frac{\mathtt{a} + \overline{D}_k + S_k(\mathtt{a},t) - \overline{D}_i}{T_i} \right\rfloor + 1 \right) \right)^+ \cdot C_i \ .$$

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399