Efficient Utilization of Bus Idle Times in CAN-based Networked Control Systems ⋆

Pau Martí * **Antonio Camacho** * **Manel Velasco** *
Mohamed El Mongi Ben Gaid **

* *Automatic Control Department, Technical University of Catalonia, Pau Gargallo 5, 08028 Barcelona, Spain (e-mail: {pau.marti,antonio.camacho.santiago,manel.velasco}@upc.edu)*
** *Technology, Computer Science, and Applied Mathematics Division, IFP, 92852 Rueil-Malmaison, France (e-mail: mongi.ben-gaid@ifp.fr)*

Abstract: This paper presents a novel approach to networked control systems (NCS) analysis and design that provides increased control performance for a set of control loops that exchange control data over the Controller Area Network (CAN). This is achieved by enabling the following functionality for each control loop: first, standard periodic messaging is guaranteed to ensure stability, and second, non-periodic additional messaging is added whenever the bus is idle in such a way that the aggregated control performance for all control loops is improved. The proposed approach, named Maximum Difference (MD) policy, is computable in a distributed manner, and is practically feasible (computationally efficient and CAN-implementable). We theoretically prove that the MD policy behaves better than static strategies. Simulation results complement the theoretical derivations and show that the MD policy outperforms static, random and Largest Error First policies.

Keywords: Networked control systems, Controller Area Network, LQR control, bandwidth management, control performance maximization.

1. INTRODUCTION

The most common design and implementation approach for NCS consists in the periodic execution of the control algorithm, which implies that control messaging is periodic (Hristu-Varsakelis et al., 2008). The benefit of this approach is that each control loop stability can be easily guaranteed at the expenses of making a static use of the bandwidth: the given messages periodicity is imposed from the control design stage regardless of the current load in the network and/or changes in the plants that are being controlled. To overcome this periodicity limitation, this paper presents a novel approach to NCS analysis and design, which, by combining periodic messaging with additional non-periodic messaging, provides increased control performance for a set of control loops that share a single CAN broadcast domain.

The baseline idea of the approach is to use at run-time the available bus idle times for applying as many control actions as possible in such a way that the aggregated control performance is improved. This is achieved by defining a policy that permits choosing and transmitting specific additional messages whenever the bus is idle in an *instantaneous* manner, without requiring to know a priori the fraction of available bandwidth. The definition of the policy guarantees that, when spare bandwidth can be consumed, each control action accommodated in the bus is chosen is such a way that, taking into account the controlled plants' dynamics, the best control performance increase among all possible choices is achieved. The definition of the policy is also inspired by the observation that certain functions such as min or max (minimum and maximum value, respectively) can be efficiently implemented in CAN (Andersson et al., 2008). Looking at CAN, the key idea is 1) to encode quantities in sensor messages identifiers, and 2) to achieve contention for the medium, in such a way that, after the operation of the bitwise arbitration phase, the desired function is accomplished. Using this principle, and under the optimal control formalism, the paper shows that the efficient utilization of bus idle times among several control loops can be formulated as a max problem that is then efficiently solved by the CAN bitwise arbitration.

The approaches introduced in Walsh et al. (2001), Yépez et al. (2003), and Anta et al. (2009) present similarities to the approach described in this paper. However, their policies are based on instantaneous metrics rather than a metric based on some sort of prediction as it is developed in this paper. Note that according to Cervin et al. (2010) the most successful approaches for allocating idle times to controllers is to apply policies based on metrics that use predictions of the expected plant dynamics rather than instantaneous metrics. In a parallel track, other works such as the approach presented by Dačić et al. (2007) focus on relaxing stability analysis conservativeness of previous works. The stability guarantees of the presented approach are given by the guaranteed periodic operation of each control loop, while the main goal is to improve aggregated control performance, while guaranteeing stability.

⋆ This work was partially supported by C3DE Spanish CICYT DPI2007-61527 and by ArtistDesign NoE IST-2008-214373.

This paper is organized as follows. First, the formalization of the NCS model and the allocation problem is addressed in Section II. Section III defines the optimization problem to be solved. Section IV reformulates this problem in order to allow its solving in a distributed manner. Section V analyzes the computational and memory demands at sensor nodes for implementing the allocation policy. Finally, simulation results showing the effectiveness of the proposed approach are described in Section VI.

2. PROBLEM FORMULATION

2.1 Networked Control System Model

The networked control system considered in this paper consists of $i = 1 \dots n$ control loops, each one formed by a sensor, a controller and an actuator implemented in physically separated nodes and sharing a single CAN bus (single broadcast domain) to exchange the control data required for each control loop operation or *job*. In terms of bandwidth utilization, each control job requires transmitting the sample in the sensor message and the control signal in the control message. In addition, other nodes, also use the CAN network to exchange other non-control data.

Henceforth, to keep the notation simple, the loop index i is dropped in the plant, controller, and cost whenever not required. Each plant is described by a continuous-time linear system

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t), \\ y(t) &= C\,x(t), \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{r\times r}$, $B \in \mathbb{R}^{r\times 1}$ and $C \in \mathbb{R}^{1\times r}$ are the system, input and output matrices, respectively, and where $x \in \mathbb{R}^{r\times 1}$ is the plant state, $u \in \mathbb{R}$ the system input (control signal), and $y \in \mathbb{R}$ the system output (plant response). Each control loop performance is measured by an infinite-horizon continuous-time quadratic cost function

$$J_c = \int_0^\infty x(t)^T Q_{c1} x(t) + u(t)^T Q_{c2} u(t) dt, \tag{2}$$

where $Q_{c1} \in \mathbb{R}^{r\times r}$ and $Q_{c2} \in \mathbb{R}$ are constant weighting matrices over the integration time. As an example of notation, for a i^{th} control loop, cost (2) is referred as $J_{c,i}$.

2.2 Mandatory and Optional Control Jobs

Within each control loop, two types of control jobs can be distinguished depending on whether they are guaranteed or not. Each ith control loop has a guaranteed constant sampling period h_i, which means that at least, with a given periodicity, the plant is sampled and the control signal is computed and updated (and held constant until a next control job executes). These control jobs are called *mandatory control jobs*. Note that mandatory control jobs of different control loops can have different guaranteed sampling periods.

In addition, each control loop can make use of additional samples between mandatory control jobs to further update the control signal. These samples are taken at a given (fast) rate, that for simplification purposes, is specified by a short sampling period h_s common to all networked control loops and specified in such a way that $\forall i = 1 \dots n$, $h_i = a_i h_s$ with $a_i \in \mathbb{N} - \{0\}$. Each control job using an additional sample is called an *optional control job* because its execution depends on the available bandwidth. If executed, a new control signal is computed and updated (and held constant until a next control job executes).

The controller gain of mandatory control jobs is designed using periodic linear quadratic (LQR) control theory for the given h_i period, which ensures stability and minimal acceptable performance. The design of gains for optional control jobs is performed borrowing the accelerable control design approach presented in Ben Gaid et al. (2008), which ensures that the more executions of additional optional control jobs within mandatory control jobs, the better will be the control performance (note that using constant feedback gains (or the same as those of the mandatory jobs) does not necessarily ensure this property, as illustrated in Ben Gaid et al. (2008)). Let k denote a control job execution and let t_k denote the time at which the k-control job executes, that is, it samples the state, $x(t_k)$, and computes and updates the control signal, $u(t)$. For any k-optional control job executing at time t_k, and taking into account that the next mandatory job will execute at t_{k_m}, with $t_k < t_{k_m}$, the control signal is given by

$$u(t) = u(t_k) = -L(t_k, t_{k_m}) x(t_k) \quad t \in [t_k, t_{k+1}), \tag{3}$$

where $L(\cdot)$ is designed using 1) equations (22) and (23) in Ben Gaid et al. (2008), and 2) considering that the control signal should be held from t_k to t_{k_m}. However, as explicitly expressed in (3), the control signal for the k-job is held constant until the next control job executes at time t_{k+1}, which can be the next mandatory job or an additional optional control job executing after the k-job. Note that one or more optional jobs can be executed between consecutive mandatory control jobs.

2.3 Discrete Cost and Notation

Consider the continuous cost function (2). The equivalent cost J_d at discrete time instants, given by the execution time of control jobs of interest, is obtained by integrating (2) over intervals given by every two consecutive control jobs. J_d is defined as

$$J_d = \sum_{k=0}^{\infty} J|_{t_k}^{t_{k+1}}(x(t_k), u(t_k)), \tag{4}$$

where k and $k+1$ denote two consecutive control jobs, $J|_{t_k}^{t_{k+1}}(\cdot)$ denotes the cost during the time interval $[t_k, t_{k+1})$, $x(t_k)$ denotes the sampled state, and $u(t_k)$ the updated control signal generated by the k job, and where

$$J|_{t_k}^{t_{k+1}}(x(t_k), u(t_k)) = \int_{t_k}^{t_{k+1}} x^T(t) Q_{c1} x(t) + u^T(t) Q_{c2} u(t) dt. \tag{5}$$

Noting that, as shown in (Åström et al., 1997)

$$x(t_k + \tau) = \Phi(\tau) x(t_k) + \Gamma(\tau) u(t_k),$$

where

$$\tau \in [t_k, t_{k+1}) \ , \ \ \Phi(\tau) = e^{A\tau} \ , \ \ \Gamma(\tau) = \int_0^\tau e^{As} ds B \ ,$$

and reminding that the control signal is held constant over consecutive control jobs, the cost over two consecutive control jobs (5) in discrete form is (Åström et al., 1997)

$$J|_{t_k}^{t_{k+1}}(x(t_k),u(t_k)) = x^T(t_k)Q_1|_{t_k}^{t_{k+1}}x(t_k) + 2x^T(t_k)Q_{12}|_{t_k}^{t_{k+1}}u(t_k) + u^T(t_k)Q_2|_{t_k}^{t_{k+1}}u(t_k) \quad (6)$$

where

$$Q_1|_{t_k}^{t_{k+1}} = \int_{t_k}^{t_{k+1}} (\Phi^T(\tau)Q_{c1}\Phi(\tau))d\tau, \quad (7)$$

$$Q_{12}|_{t_k}^{t_{k+1}} = \int_{t_k}^{t_{k+1}} (\Phi^T(\tau)Q_{c1}\Gamma(\tau))d\tau, \quad (8)$$

$$Q_2|_{t_k}^{t_{k+1}} = \int_{t_k}^{t_{k+1}} (\Gamma^T(\tau)Q_{c1}\Gamma(\tau) + Q_{c2})d\tau. \quad (9)$$

Cost (6) can be written in compact form as

$$J|_{t_k}^{t_{k+1}}(x(t_k),u(t_k)) = \begin{bmatrix} x(t_k) \\ u(t_k) \end{bmatrix}^T Q|_{t_k}^{t_{k+1}} \begin{bmatrix} x(t_k) \\ u(t_k) \end{bmatrix}, \quad (10)$$

where

$$Q|_{t_k}^{t_{k+1}} = \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^T & Q_2 \end{bmatrix}\Bigg|_{t_k}^{t_{k+1}}. \quad (11)$$

Again, as an example of notation, for the i^{th} control loop, cost (4) can be referred as $J_{d,i}$. To improve paper readability, a short notation is introduced. First of all, for a given i^{th} control loop, costs between consecutive control jobs k and $k+1$ as in (6) or (10) will be denoted by $J|_{t_k}^{t_{k+1}}$. Hence,

$$J_i|_{t_k}^{t_{k+1}} \equiv J_i|_{t_k}^{t_{k+1}}(x(t_k),u(t_k)). \quad (12)$$

Second, for a given i^{th} control loop, considering the time interval $[t_{k_m},\infty)$, the cost achieved by executing only mandatory jobs starting to execute at t_{k_m} is denoted by

$$J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_p) = \sum_{j=k_m}^{\infty} J_i|_{t_j}^{t_j+h_i}(x(t_j),u(t_j)), \quad (13)$$

where t_p denotes the time when the previous job execution took place (if any) before the execution time of the first mandatory job under evaluation, t_{k_m}. Hence, 1) it may indicate that no job executed previously, $t_p = -$, or 2) it may correspond to the execution time of the previous mandatory control job, $t_p = t_{k_m} - h$, or 3) it may correspond to the execution time of a previous k^{th} optional control job, $t_p = t_k$. The t_p parameter will determine which state $x(t_{k_m})$ for the first term of the summand (when $t_j = t_{k_m}$) has to be considered.

2.4 Problem to Be Solved

The NCS model assumed in this paper considers n control loops, each one with a plant in the form of (1) whose control signal is given by (3) for both mandatory and optional control jobs (if possible), and whose control performance is evaluated via (4)-(6). For the n control loops, at any time different than mandatory control jobs execution times, provided that bandwidth is available for executing an optional control job, the problem to be solved, namely *decision*, is to which control loop the optional control job should be allocated to, such that the overall accumulated cost

$$J_{d,total} = \sum_{i=1}^{n} J_{d,i} \quad (14)$$

is minimized. Note that (14) accounts for both mandatory and optional control jobs.

The desired solution to the *decision* problem should have the form of a max or min problem. This would facilitate its implementation in CAN. This requires addressing three sub-problems:

(1) As stressed in the introduction, successful approaches for allocating idle times to controllers demand defining policies based on metrics that use predictions of the expected plant dynamics (rather than instantaneous metrics such as a function of the current plants error). However, such policies require knowing the activation pattern of all future control jobs, which can not be known in the scenario considered in this paper. Therefore, a first problem to be solved, namely *definition*, is to define a suitable policy/metric in the form of a max or min problem for deciding which optional control job must be executed any time such that (14) is minimized.
(2) The allocation policy must be defined considering that its implementation will be performed distributedly, using the bitwise arbitration phase of CAN. Hence, the quantities encoded in the message identifiers participating in the policy implementation must be computed using local parameters to the sending node. Therefore, the second problem to be solved, namely *distribution*, is to redefine the allocation policy in such a way that this restriction is fulfilled.
(3) The computation of the quantities to be encoded in messages identifiers must be easy to implement in the sense of reducing the memory/processor requirements as much as possible. Therefore, the third problem to be solved, namely *feasibility*, is to provide a feasible method for computing the required quantities minimizing resource demands within the sample time.

3. MINIMUM COST POLICY

This section addresses the *definition* problem. Since the times at which optional control jobs will execute is not known a priori (because their possible execution depends on the run-time available bandwidth), the policy for allocating bus idle times to control loops is defined in terms of what can be known when the bus is idle: the future execution times of mandatory jobs (the periodic ones) plus the possible benefit of executing an additional control job. Specifically, from the current time to ∞, n costs are evaluated. Each one corresponds to the total aggregated cost that would be achieved if the optional control job would be allocated to a particular i^{th} control loop while considering all mandatory control jobs. From the n costs, the minimum indicates to which control loop is more beneficial the optional control job allocation in terms of decreasing aggregated cost. This is formalized next.

Definition 1. [Cost with optional control job] For a given i^{th} control loop, at any given current time t_k such that

$t_{k-1} < t_k < t_{k_m}$, where t_{k-1} denotes the previous (mandatory or optional) job execution time, and t_{k_m} denotes the next mandatory job execution time, the cost that would be achieved in the time interval $[t_{k-1}, \infty)$ by executing an additional k-optional control job at t_k is

$$c_i^{opt}(t_k) = J_i|_{t_{k-1}}^{t_k} + J_i|_{t_k}^{t_{k_m}} + J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_k). \quad (15)$$

Definition 2. [Cost without optional control job] Under the same conditions given in definition 1, the cost that would be achieved in the time interval $[t_{k-1}, \infty)$ by not executing an additional k-optional control job at t_k is

$$c_i^{Nopt}(t_k) = J_i|_{t_{k-1}}^{t_{k_m}} + J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_{k-1}). \quad (16)$$

Definition 3. [Total cost with 1 optional control job] For a set of n control loops, the total aggregated cost as in (14) that would be achieved in the time interval $[t_{k-1}, \infty)$ by executing in the i^{th} control loop an additional k-optional control job at time t_k such that $t_{k-1} < t_k < t_{k_m}$ is

$$c_i^{total}(t_k) = c_i^{opt}(t_k) + \sum_{j=1, j\neq i}^{n} c_j^{Nopt}(t_k), \quad (17)$$

where t_{k-1} and t_{k_m} are defined as in Definition 1.

Definition 4. [Minimum Cost policy - MC] Considering n control loops, each one with a plant in the form of (1) whose control signal is given by (3) for both mandatory and optional control jobs, and whose control performance is evaluated via (4)-(6), provided that each optional control job can be only allocated to one control loop at a time if bandwidth is available, at any given current time t_k such that $t_{k-1} < t_k < t_{k_m}$, the k^{th}-optional control job is allocated to the i^{th}-control loop given by

$$\arg\min_i \{c_i^{total}(t_k) \mid i = 1, \ldots, n\}, \quad (18)$$

where t_{k-1} and t_{k_m} are defined as in Definition 1.

By encoding in each sensor message identifier each argument $c_i^{total}(t_k)$ of the min function in (18), in bus contention, the bitwise arbitration of CAN would produce the desired result: give access to the message encoding the minimum cost. However for each i^{th} control loop, each argument depends on local parameters such as $c_i^{opt}(t_k)$ but also on non-local parameters such as all $c_j^{Nopt}(t_k)$, $j \neq i$. In other words, the i^{th} sensor would require to have available all other sensor readings in order to compute its argument.

4. MAXIMUM DIFFERENCE POLICY

This section shows that the MC policy can be transformed into an equivalent policy that preserves the same logics, i.e., it identifies the same i^{th} control loop, and each quantity encoded in each sensor message priority depends only on local parameters to the sensor, thus solving the *distribution* problem. The idea is to assess which control loop provides the maximum difference between the total aggregated cost without optional control job and each of the aggregated costs achieved when allocating the optional control job to a particular i^{th} control loop. In this case, from the n cost differences, the maximum indicates to which control loop is more beneficial the optional control job allocation. This idea is formalized next.

Definition 5. In the time interval $[t_{k-1}, \infty)$, the cost difference that for the i^{th} control loop would suppose to execute the k-optional control job is defined from (15) and (16) as

$$d_i(t_k) = c_i^{Nopt}(t_k) - c_i^{opt}(t_k). \quad (19)$$

Theorem 6. Under the assumptions taken in the MC policy (definition 4), it holds that

$$\arg\min_i \{c_i^{total}(t_k) \mid i = 1, \ldots, n\} = \arg\max_i \{d_i(t_k) \mid i = 1, \ldots, n\}. \quad (20)$$

Proof. Starting with the right hand side term of the equality (20), and considering that

$$\arg\max_i \{d_i(t_k) \mid i = 1, \ldots, n\} = \arg\min_i \{-d_i(t_k) \mid i = 1, \ldots, n\}$$

holds, the definition of $d_i(t_k)$ given in (19) can be placed into the right hand side of the previous equality and rearranged as

$$\arg\min_i \{c_i^{opt}(t_k) - c_i^{Nopt}(t_k) \mid i = 1, \ldots, n\}. \quad (21)$$

Observing that adding a constant to the arguments of a arg min function does not alter its result, the expression $\sum_{j=1}^{n} c_j^{Nopt}(t_k)$ is added to the arguments of the min function in (21), which results in

$$\arg\min_i \{c_i^{opt}(t_k) - c_i^{Nopt}(t_k) + \sum_{j=1}^{n} c_j^{Nopt}(t_k) \mid i = 1, \ldots, n\}.$$

Finally, it can be observed that by simplifying in the previous equation the term $c_i^{Nopt}(t_k)$ from the summand, the arguments of the min function reduces to the definition of $c_i^{total}(t_k)$ given in (17). Therefore, the left hand side of (20) is obtained.

Corollary 7. [Maximum Difference policy - MD] It follows that the MC policy given in definition 4 can be rephrased by changing equation (18) to

$$\arg\max_i \{d_i(t_k) \mid i = 1, \ldots, n\}. \quad (22)$$

In this case, the policy is named MD policy.

It can be observed that the arguments of the max function in (22), $d_i(t_k)$, are much simpler that the arguments of the min function in (18), $c_i^{total}(t_k)$. But more important, it can also be observed that each cost difference $d_i(t_k)$ can be computed at run-time by each i^{th} closed-loop sensor node without requiring data from the other sensor nodes. Therefore, for the MD policy, if all sensor nodes encode in their optional sensor messages' identifiers the 1's complement of their specific $d_i(t_k)$, in bus contention, the bitwise arbitration phase will allow bus access to the control loop providing the maximum difference, that is, providing the minimum aggregated cost. Hence, the appropriate optional control job will start executing.

5. MEMORY/PROCESSOR REQUIREMENTS

The implementation of the MD policy requires computing the cost difference $d_i(t_k)$ (19) at each sensor node. This section presents a method for computing $d_i(t_k)$ efficiently in terms of both computational and memory demands, thus solving the *feasibility* problem.

The cost difference $d_i(t_k)$ (19) is the difference between sub-costs $c_i^{Nopt}(t_k)$ (16) and $c_i^{opt}(t_k)$ (15), where each one has a term which is an infinite summand of the form $J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_p)$ as defined in (13). This infinite summand reflects the cost achieved by executing, from a given time t_{k_m}, only mandatory control jobs. Noting that the gain for mandatory control jobs are designed using standard LQR control theory, and noting also that mandatory control jobs execute periodically with a sampling interval given by h, terms $J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_p)$ can be computed as (Åström et al., 1997)

$$J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_p) = x_{t_p}^T(t_{k_m})S_i(h)x_{t_p}(t_{k_m}), \tag{23}$$

where $S_i(h)$ is the solution to the algebraic Riccati equation, and $x_{t_p}(t_{k_m})$ in (23) varies depending on the previous job execution time with respect to the first mandatory job executing at t_{k_m}. Hence, $x_{t_p}(t_{k_m})$ can be computed as

$$x_{tp}(t_{k_m}) = \Phi_{cl}|_{t_p}^{t_{k_m}}(t_{k_m} - t_p)x(t_p), \tag{24}$$

where

$$\Phi_{cl}|_{t_a}^{t_b} = \Phi(t_b - t_a) - \Gamma(t_b - t_a)L(t_a, t_{k_m}). \tag{25}$$

Using this simplification, it can be easily counted that at each sensor node, and for each optional control job, a total of 24 matrix operations have to be performed to compute $d_i(t_k)$ while requiring to have stored in sensor memory a number of $4(\frac{h}{h_s} - 1) + 1$ matrices ($\frac{h}{h_s} - 1$ is the number of optional control jobs between mandatory control jobs), which may still not be feasible.

The computation of $d_i(t_k)$ (19) can be further simplified if it is computed over $[t_k, \infty)$ rather than $[t_{k-1}, \infty)$. This can be done because the plant evolution over the time interval $[t_{k-1}, t_k)$ is identical when computing $c_i^{Nopt}(t_k)$ and $c_i^{opt}(t_k)$. In addition, each sensor node can memorize the last updated control input, namely $u(t_k^-)$ (regardless of whether it was updated by a mandatory or an optional control job) from the previous control message. Hence, $u(t_k^-)$ can be used instead of requiring keeping track of t_p in (23). This is formalized as follows.

Proposition 8. With $u(t_k^-)$ being the value of the previous control update for a given optional instant t_k, the maximum difference $d_i(t_k)$ in (19) can be alternatively computed as

$$d_i(t_k) = \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}^T \mathbb{Q}(t_k, t_{k_m}) \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}, \tag{26}$$

where

$$\mathbb{Q}(t_k, t_{k_m}) = \mathbb{Q}^{Nopt}(t_k, t_{k_m}) - \begin{bmatrix} \mathbb{Q}^{opt}(t_k, t_{k_m}) & 0 \\ 0 & 0 \end{bmatrix} \tag{27}$$

with

$$\mathbb{Q}^{Nopt}(t_k, t_{k_m}) = Q|_{t_k}^{t_{k_m}} + [\Phi(t_k, t_{k_m}), \Gamma(t_k, t_{k_m})]^T S_i(h) [\Phi(t_k, t_{k_m}), \Gamma(t_k, t_{k_m})]$$

and

$$\mathbb{Q}^{opt}(t_k, t_{k_m}) = \begin{bmatrix} I \\ -L(t_k, t_{k_m}) \end{bmatrix}^T Q|_{t_k}^{t_{k_m}} \begin{bmatrix} I \\ -L(t_k, t_{k_m}) \end{bmatrix} + \Phi_{cl}|_{t_k}^{t_{k_m}}(t_{k_m} - t_k)^T S_i(h) \Phi_{cl}|_{t_k}^{t_{k_m}}(t_{k_m} - t_k). \tag{28}$$

Proof. By noting that the term $J_i|_{t_{k-1}}^{t_{k_m}}$ for computing $c_i^{Nopt}(t_k)$ (16) can be rewritten as follows

$$J_i|_{t_{k-1}}^{t_{k_m}}(x(t_{k-1}), u(t_{k-1})) = J_i|_{t_{k-1}}^{t_k}(x(t_{k-1}), u(t_{k-1})) + J_i|_{t_k}^{t_{k_m}}(x(t_k), u(t_{k-1})), \tag{29}$$

the cost difference (19) directly transforms to

$$d_i(t_k) = c_i^{Nopt}(t_k) - c_i^{opt}(t_k) \tag{30}$$

$$= J_i|_{t_k}^{t_{k_m}}(x(t_k), u(t_{k-1})) + J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_{k-1}) \tag{31}$$

$$- (J_i|_{t_k}^{t_{k_m}}(x(t_k), u(t_k)) + J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_k)). \tag{32}$$

In the previous equation, (31) corresponds to $c_i^{Nopt}(t_k)$ and (32) corresponds to $c_i^{opt}(t_k)$.

Focusing in (31), the term $J_i|_{t_k}^{t_{k_m}}(x(t_k), u(t_{k-1}))$ can be also written as $J_i|_{t_k}^{t_{k_m}}(x(t_k), u(t_k^-))$ using the known value $u(t_k^-)$. In addition, by using (23), its right summand is defined as

$$J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_{k-1}) = x_{t_{k-1}}^T(t_{k_m})S_i(h)x_{t_{k-1}}(t_{k_m}) \tag{33}$$

where $x_{t_{k-1}}(t_{k_m})$ can be computed based only on $x(t_k)$ and $u(t_k^-)$ as

$$x_{t_{k-1}}(t_{k_m}) = \Phi(t_k, t_{k_m})x(k) + \Gamma(t_k, t_{k_m})u(t_k^-) = [\Phi(t_k, t_{k_m}), \Gamma(t_k, t_{k_m})] \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}. \tag{34}$$

By considering (33) and (34), and by noting that using the compact notation (10) the left summand of (31) is defined as

$$J_i|_{t_k}^{t_{k_m}} = \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}^T Q|_{t_k}^{t_{k_m}} \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}, \tag{35}$$

expression (31) can be computed as

$$\begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}^T \mathbb{Q}^{Nopt}(t_k, t_{k_m}) \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}, \tag{36}$$

where $\mathbb{Q}^{Nopt}(t_k, t_{k_m})$ is given in (28).

Focusing in (32), its left summand using the compact notation (10) is defined as

$$J_i|_{t_k}^{t_{k_m}} = \begin{bmatrix} x(t_k) \\ u(t_k) \end{bmatrix}^T Q|_{t_k}^{t_{k_m}} \begin{bmatrix} x(t_k) \\ u(t_k) \end{bmatrix}, \tag{37}$$

which can be rewritten using $u(t_k) = -L(t_k, t_{k_m})x(t_k)$ as

$$J_i|_{t_k}^{t_{k_m}} = x(t_k)^T \begin{bmatrix} I \\ -L(t_k, t_{k_m}) \end{bmatrix}^T Q|_{t_k}^{t_{k_m}} \begin{bmatrix} I \\ -L(t_k, t_{k_m}) \end{bmatrix} x(t_k). \tag{38}$$

Considering (23) and (24) with $t_p = t_k$, the right summand of (32) is

$$J_{d,i}^{man}|_{t_{k_m}}^{\infty}(t_k) = x_{t_k}^T(t_{k_m})S_i(h)x_{t_k}(t_{k_m}), \tag{39}$$

where $x_{t_k}(t_{k_m})$ can be computed as

$$x_{t_k}(t_{k_m}) = \Phi_{cl}|_{t_k}^{t_{k_m}}(t_{k_m} - t_k)x(t_k). \tag{40}$$

Hence, using (38), (39) and (40), expression (32) can be computed as

$$x(t_k)^T \mathbb{Q}^{opt}(t_k, t_{k_m})x(t_k) \tag{41}$$

or alternatively as

$$\begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}^T \begin{bmatrix} \mathbb{Q}^{opt}(t_k, t_{k_m}) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t_k) \\ u(t_k^-) \end{bmatrix}, \tag{42}$$

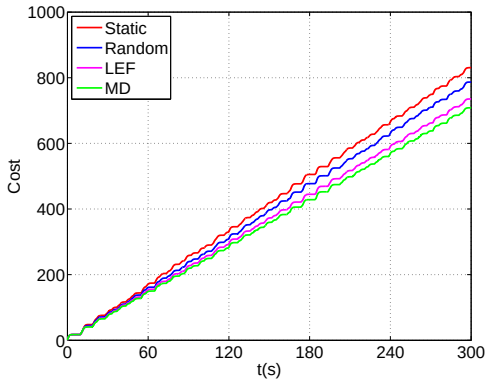

Fig. 1. Performance evaluation

where $\mathbb{Q}^{opt}(t_k, t_{k_m})$ is given in (28).

Substituting (36) and (42) into (30), the definition of $d_i(t_k)$ given in (26) and (27) is obtained.

By simple inspection of (26), the required number of matrix operations at each sensor node for each optional control job reduces to 3. And the number of matrices to be stored in memory reduces to $\frac{h}{h_s} - 1$, which correspond to the matrices $\mathbb{Q}(t_k, t_{k_m})$ to be precomputed for all t_k than can occur between two consecutive mandatory jobs.

6. SIMULATIONS

Three networked control loops, each one controlling a double integrator system

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t), \end{aligned} \tag{43}$$

have been simulated under different policies using the TrueTime simulator presented by Henriksson et al. (2002). Each plant is subject to random set-point changes of ± 1 that occur every 10s (in average) in a run of 300s. The sampling period for mandatory control jobs for the three loops is the same and equal to $h = 1$s, and the sampling period for optional control jobs is $h_s = h/3$. For each loop under the MD policy, controller gains for both types of control jobs have been designed as indicated in Section 2.2 where the weighting matrices Q_{c1} and Q_{c2} of the continuous cost function given in (2) are the identity.

The MD policy has been compared to three different policies. The first one, named *static*, corresponds to the case where the three control loops only run mandatory control jobs, which represents the standard periodic approach. Since any execution of an optional control job is able to decrease the cost, the MD policy is also compared to a *random* policy that corresponds to the case where optional control jobs are randomly allocated to control loops. The third one in the performance evaluation is the *Largest Error First (LEF)* policy presented by Yépez et al. (2003). In LEF, control gains for both mandatory and optional control jobs are constant and designed using LQR design for a sampling period of $h = 1$s, and bus idle times are always assigned to the plant with largest error.

Figure 1 shows the simulation results where the x-axis is simulation time and the y-axis is the accumulated cost of the three plants calculated as $J_{eval} = \sum_{i=1}^{3} J_{c,i}$.As can be seen in Figure 1, the MD policy provides the best cost (lowest curve). The static policy gives the worst cost because it only executes mandatory control jobs. The curve of the random policy is always above the curve of the MD policy. This indicates that the MD policy developed in this paper is effective in terms of allocating bus idle times to control loops. And finally, the better performance numbers of the MD policy with respect to the LEF policy shows that increased performance can be achieved by allocating bus idle times using a metric based on predictions of the plants' dynamics rather than using an instantaneous metric as in the case of LEF. Future work will also consider computational complexity and its effect on the performance numbers.

7. CONCLUSIONS

For a set of networked control loops, this paper has presented the MD policy that permits to chose at run time which loop should execute an additional control job each time the bus is idle. It has been shown that the policy can be efficiently implemented in CAN, and its feasibility in terms of resource demands has been discussed. Simulations showed that the MD policy provides the best performance compared to previous and/or alternative policies.

REFERENCES

B. Andersson, N. Pereira, W. Elmenreich, E. Tovar, F. Pacheco, and N. Cruz. A Scalable and efficient approach for obtaining measurements in CAN-based control systems. *IEEE Transactions on Industrial Informatics,* vol.4, no.2, pp.80-91, May 2008.

A. Anta and P. Tabuada. On the benefits of relaxing the periodicity assumption for networked control systems over CAN. *Real Time Systems Symposium*, Dec. 2009.

K.J. Åström, and B. Wittenmark. *Computer-controlled systems.* Third Edition, Prentice-Hall, 1997.

M. Ben Gaid, D. Simon, and O. Sename. A Design methodology for weakly-hard real-time control *17th IFAC World Congress on Automatic Control*, Seoul, Korea, July 2008.

A. Cervin, M. Velasco, P. Martí, and A. Camacho. Optimal on-line sampling period assignment: theory and experiments. *IEEE Transactions on Control Systems Technology*, accepted for publication, June 2010.

D. B. Dačić and D. Nešić. Quadratic stabilization of linear networked control systems via simultaneous protocol and controller design. *Automatica*, v. 43, n.7, pp. 1145-1155, Jul. 2007.

D. Henriksson, A. Cervin, and K.-E. Årzén. TrueTime: Simulation of control loops under shared computer resources. *15th IFAC World Congress*, 2002.

D. Hristu-Varsakelis and W. S. Levine. *Handbook of networked and embedded control systems* Birkhäuser Boston, June, 2008.

G.C. Walsh, and Y. Hong. Scheduling of networked control systems. *IEEE Control Systems Magazine,* vol. 21, n. 1, pp. 57-65, Feb. 2001

J. Yépez, P. Martí and J.M. Fuertes. Control loop scheduling paradigm in distributed control systems. *29th Annual Conference of the IEEE Industrial Electronics Society*, Roanoke, USA, November, 2003.