On the benefits of relaxing the periodicity assumption for networked control systems over CAN

Adolfo Anta and Paulo Tabuada
Dept. of Electrical Engineering
University of California, Los Angeles
E-mail: {adolfo,tabuada}@ee.ucla.edu

***Abstract*—A vast majority of control systems require the use of networks for the communication between the different agents: sensors, controllers, and actuators. The existing paradigm regards the messages, between sensors and controllers and between controllers and actuators, as periodic. Although this strategy facilitates the analysis and implementation, it leads to a conservative usage of the communication bandwidth. Based on previous work by the authors, an aperiodic strategy is proposed in this paper for the dynamic allocation of bandwidth according to the current state of the plants and the available resources. The case of control loops closed over Controller Area Networks (CANs) is discussed in detail and illustrated on a train car.**

I. Introduction

In distributed environments, control systems are no longer implemented using point-to-point networks, but rather through a shared bus where controllers, sensors, and actuators exchange data. Under this setup, control systems designers usually impose stringent constraints on the timing of the network in order to guarantee a desired performance. One of the most common assumptions for networked control systems consists in the periodic execution of the control algorithm. This requirement leads to a conservative usage of resources, since messages are sent through the network at the same rate regardless of the current load in the network and the behaviour of the system being controlled. Moreover, this rate is selected in order to provide performance guarantees under worst case conditions even if these only rarely occur.

To overcome this periodicity assumption several researchers, mainly from the real-time systems community, have applied rate adaptation techniques, where the period is selected according to the current state of the plant and the available bandwidth. Such strategies have already been widely studied in the context of shared processors (e.g. [SLSS96]) and are currently being extended to networked control systems. While in the case of shared processors information is centralized and the scarce resource is the computation time, in networks the data is in general only available locally and the constrained resource is the network bandwidth. The rate adaptation techniques suffer from two main drawbacks. First, the starting point of all these studies is the definition of performance as a monotonic function of the rate: the larger the rate, the better the performance, which does not need to hold in general [Bam03]. Second, quadratic or exponential performance functions are commonly used [AHBLP06], which seems to be a sensible decision but far from optimal and with no theoretical support.

A different approach, in the line of event-triggered techniques, consists in dynamic bandwidth allocation granting more resources to those control loops with larger errors ([VFL$^+$04], [WY01]). Recent work has also explored the use of event-triggered control techniques in the context of wireless networks to reduce the communication between sensors, controllers and actuators ([WL09], [MJT08]). All such studies have been carried only for linear control systems.

In this paper we advocate for an aperiodic implementation of control loops, as it leads to a more efficient allocation of resources without jeopardizing performance. Drawing inspiration from event-triggered control, the authors previously defined in [AT08b] the real-time requirements of a control loop as a function of the current state of the system being controlled. Such strategy, known as self-triggered control, computes the next time the input needs to be updated based on the last measurement of the system state. The self-triggered conditions reveal the inherent interplay between performance and resource allocation. The main purpose of this paper is to show how self-triggered techniques are also beneficial in reducing the bandwidth usage in networked control systems. To consider a practical case, we study the behaviour of control loops over a CAN network and show how to implement the proposed resource-aware strategy. First we perform a schedulability analysis for self-triggered controllers. Second, a dynamic bandwidth allocation strategy that exploits the benefits of the self-trigger strategy is proposed. The algorithm can be easily extended to other dominant network protocols, such as TTCAN, as explained in section VI. The study carried on herein is applicable to linear and nonlinear control systems. Our approach also facilitates the analysis of other network effects such as delays and packet dropouts. This paper constitutes the next step in our journey to promote the benefits of self-triggered control for embedded control systems.

II. Problem statement

The starting point is a control system:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \qquad \mathbf{x}(t) \in \mathbb{R}^n,\ \mathbf{u}(t) \in \mathbb{R}^m \tag{II.1}$$

for which a feedback controller:

$$\mathbf{u} = k(\mathbf{x}) \tag{II.2}$$

has been designed, rendering the closed loop system $\dot{\mathbf{x}} = f(\mathbf{x}, k(\mathbf{x}))$ asymptotically stable. We use $\mathbf{x}$ for the state trajectory, $\mathbf{u}$ for the input trajectory and x to denote a point

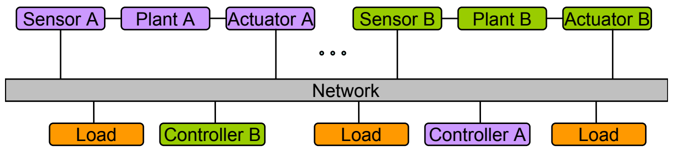

Fig. 1. Network topology.

in the state space. Whenever it is needed to emphasize the initial condition for the trajectory we write $\mathbf{x}(t, x_0)$, where $\mathbf{x}(0, x_0) = x_0$. Controller (II.2) is implemented in a digital platform that is linked to the sensor and actuator through a network, shared by several control loops in addition to other nodes, as shown in Figure 1. Whenever the feedback law needs to be recomputed, the sensor measures the state x at time instant t_i and sends a message (called sensor message) with the value of $\mathbf{x}(t_i) = x$ to the controller node. Upon the arrival of the sensor message, the controller node computes the control law $\mathbf{u}(t_i) = k(\mathbf{x}(t_i))$. Once the computation is finished, the controller node sends a message (called control message) to the actuator with the new input $\mathbf{u}(t_i)$ that needs to be applied to the plant. The actuator value will be kept constant until the actuator node receives a new message from the controller.

The problems we tackle in this paper can now be posed as follows.

- *When shall the controller and the sensor send messages through the network so that a desired performance is guaranteed?*
- *Is it possible, in case of overload or scarce available bandwidth, to reduce the number of sensor and control messages and still guarantee some level of performance?*
- *How do we allocate dynamically spare bandwidth to improve performance of the control loops or other nodes connected to the network?*

To tackle these issues, we revisit two recent implementation paradigms that lead to a more efficient utilization of the available resources.

III. Event and Self-triggered stabilization

We are interested in defining a sequence of transmission times $\{t_i\}_{i=1,2,\dots}$ for the sensor and control messages that guarantees stability and desired performance, while saving resources. Although the results of this paper apply to nonlinear systems, we shall review the event-triggered stabilization in a linear context for simplicity of presentation. In the linear case, the control system defined in (II.1) becomes:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{III.1}$$

and is asymptotically stabilized by a linear feedback:

$$\mathbf{u} = K\mathbf{x} \tag{III.2}$$

In a real system, the input cannot be updated continuously as described in (III.2) but only at discrete time instants t_i. Traditionally the input is updated periodically, *i.e.* , $t_{i+1} - t_i = T$ for all $i \in \mathbb{N}_0$, and where the period T is selected in order to guarantee stability (and a desired performance) under all possible operating conditions. This approach represents a conservative implementation since the choice of T is based on a worst-case scenario. Indeed, it seems more natural to update the input only when something significant occurs in the system.

A. Event-triggered stabilization

To determine the time instants when the input needs to be updated (*i.e.*, when messages need to be sent over the network), we study the dynamics of the system under a sample-and-hold implementation, where the plant input $\mathbf{u} = K\mathbf{x}(t_i)$ is kept constant until a new message is received by the actuator node. We first define an auxiliary variable $\mathbf{e}(t)$ representing the difference between the sampled state and the current state of the system:

$$\mathbf{e}(t) = \mathbf{x}(t_i) - \mathbf{x}(t) \qquad \forall t \in [t_i + \Delta_i, t_{i+1} + \Delta_{i+1}[$$

where t_i represents the time instants when the state is measured and $\Delta_i > 0$ denotes the delay between measurement and actuation. Using this variable we can rewrite the dynamics of our system as:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + BK\mathbf{x}(t_i) \\ &= (A + BK)\mathbf{x}(t) + BK\mathbf{e}(t)\end{aligned} \tag{III.3}$$

Since (III.2) is a stabilizing controller, it is well known from control theory [Kha02] that there exists a Lyapunov function V satisfying:

$$\dot{V} \le -a|x|^2 + b|x||e| \qquad a, b > 0 \tag{III.4}$$

where $|\cdot|$ denotes the Euclidean norm. If we restrict e to satisfy:

$$b|e| \le \sigma\, a|x| \tag{III.5}$$

the dynamics of V is bounded by:

$$\dot{V} \le (\sigma - 1)a|x|^2$$

thus guaranteeing that V decreases provided that $\sigma < 1$. In the context of nonlinear systems, equation (III.4) becomes:

$$\dot{V} \le -\alpha(|x|) + \gamma(|e|) \tag{III.6}$$

where α and γ are strictly increasing continuous functions with $\alpha(0) = \gamma(0) = 0$; and (III.5) is replaced by:

$$\gamma(|e|) \le \sigma\alpha(|x|) \tag{III.7}$$

to preserve stability of the control loop. Inequality (III.5) can be enforced by executing the control task at the time instants t_i whenever:

$$|\mathbf{e}(t_i)| = \sigma\frac{a}{b}|\mathbf{x}(t_i)| \tag{III.8}$$

Equality (III.8) generates a sequence of instants at which the input u has to be updated in order to guarantee stability. This strategy leads to a lower number of messages than the conservative periodic task model, since communication between sensor, controller and actuator only occur when it is indeed required. In the context of real-time scheduling for microprocessors, event-triggered techniques lead to a lower number of executions for the control task. In particular, previous examples analyzed by the authors in [AT08a] show a reduction of 87% in the number of executions.

The parameter σ represents the rate of convergence of the dynamical system (a measure of instantaneous quality-of-control) and at the same time it determines the bandwidth required by the control loop. Hence, this parameter σ represents a simple abstraction of the control performance that will facilitate the codesign of the control loop and the message scheduling in the network. Our definition of performance (rate of convergence) generalizes to nonlinear systems, in contrast with other performance indices previously used in the literature, such as the integral of the absolute value of the error (IAE), or phase and gain margins. Notice that performance is increased as σ diminishes.

B. *Self-triggered stabilization*

An event-triggered implementation based on equality (III.8) would require sending two messages (from sensor to controller and controller to actuator) right whenever (III.8) is satisfied. Even if communication delays are ignored, this strategy is not feasible since the control messages might not have the highest priority in the network (or there might be more than one control loop in the network). A better solution that we propose here is to use the current measurement of the state to decide the next time messages need to be sent, that is, a self-triggered control law. Our goal is to find a self-trigger condition that guarantees inequality (III.7) at any instant of time, i.e, a simple formula that determines a sequence of transmission times for the messages guaranteeing a desired performance for the control system. To find the sequence of instants $\{\tau_i = t_i + \Delta_i\}_{i\in\mathbb{N}}$ described by equality (III.8) we need to analyze the dynamics of the control system, that determines the evolution of the ratio $\frac{|e|}{|x|}$. The procedure is described in detail in our previous work [AT08b]. Due to space limitations, we briefly summarize the idea here:

- To derive a self-triggered condition, the transmission times of a control law should be expressed in terms of the measured state and the desired performance. We denote this time by $\tau(\mathbf{x}(t_i))$, as it depends on the last sample of the system $\mathbf{x}(t_i)$:
$$\tau(\mathbf{x}(t_i)) = \min\{t > t_i + \Delta_i : |\mathbf{e}(t, \mathbf{x}(t_i))| = \sigma\frac{a}{b}|\mathbf{x}(t, \mathbf{x}(t_i))|\} \quad \text{(III.9)}$$
Since the expressions of $\mathbf{x}$ and $\mathbf{e}$ are in general not known for nonlinear systems, it is not possible to compute $\tau(\mathbf{x}(t_i))$ in closed form. Hence, we exploit the geometry of nonlinear systems to avoid computing (III.9) using the trajectories of the system. For that matter, it is easy to derive formulas that do not determine directly the transmission times but describe instead the evolution of such times. At a particular state $\mathbf{x}(t_j)$, the transmission time $\tau(\mathbf{x}(t_j))$ is related to the transmission time at another state $\tau(\mathbf{x}(t_i))$ according to the formula:
$$\tau(\mathbf{x}(t_j)) = \chi(\mathbf{x}(t_j)) \cdot \tau(\mathbf{x}(t_i)) \quad \text{(III.10)}$$
where χ is a function that is determined by the dynamics of the control system. Hence, once the transmission time for some state $\mathbf{x}(t_i)$ is known, it is possible to deduce the transmission time for another state $\mathbf{x}(t_j)$, according to (III.10). That is, this equation allows us to obtain a sequence of transmission times once an initial transmission time is known.
- In order to apply equation (III.10) online, it is necessary to find the transmission time for the initial condition. It was shown in [Tab07] that this time can be obtained from the following equation:
$$\tau^*(\sigma) = \alpha_1 + \alpha_2 \arctan(\alpha_3 + \sigma\alpha_4) \quad \text{(III.11)}$$
where each α_i is a function of the dynamics of the system (II.1) and the controller (II.2). This time is commonly known in the control community as MATI (maximum allowable transfer interval). Other possible methods to compute such time appear in [NTC09].
- In equation (III.10), if we let $\tau(\mathbf{x}(t_j))$ be the next relative transmission time τ_j and $\tau(\mathbf{x}(t_i))$ be the initial transmission time $\tau^*(\sigma)$, we obtain the following self-triggered condition to be used online:
$$\tau_j = \chi(\mathbf{x}(t_j))\, \tau^*(\sigma), \qquad j = 1, 2, \ldots \quad \text{(III.12)}$$
Hence, the times between transmissions depend on the current state and on τ^*, which is in turn a function of σ, the control performance. The scheduler could modify online the value of τ^* to adjust for the required bandwidth or to optimize global performance, as it will be later shown. In other words, control messages should not considered to have hard deadlines, since tardiness (up to some value) does not imply instability but just a lower performance for the control system.

Notice that τ represents the time between two input updates, *i.e.*, the time between two control messages.

IV. Message scheduling over CAN

In this section we propose solutions to the three questions posed in Section II. To make the presentation concrete we focus on CAN buses. In subsection IV-C, an offline schedulability analysis for self-triggered controllers is derived, based on existing results. Subsection IV-D explains a dynamic bandwidth allocation strategy, which represents the main contribution of this paper.

A. *Overview of the CAN protocol*

The CAN (Controller Area Network) [Gmb91] protocol was established by Robert Bosch GmbH. The CAN is a serial communication protocol for distributed real-time control and automation systems. It is message oriented, based on CSMA/NBA (carrier sense medium access with nondestructive bitwise arbitration) mechanism. Each node senses the state of the bus and defers transmission of data until the bus state is idle. Each message has a uniquely assigned priority. Whenever two nodes attempt to send a message at the same time, the node transmitting the highest priority identifier gets the control of the bus. The network is usually shared by real-time and non real-time data. Non real-time should be sent whenever there is no real-time message that needs to be transmitted. The CAN has become increasingly popular due to its low cost and deterministic resolution of the contention, which helps providing guarantees of performance for the different systems interconnected through the network. Its major limitation is the maximum available bandwidth (1Mbit/s), due to the multi-master bus structure.

One of the main tasks, when designing the scheduling policy for the messages in the network, consists in the assignment of priorities to each message. Static priority assignment (such as rate monotonic or deadline monotonic) has been widely implemented, because of its simplicity. However, they constitute conservative solutions that do not take full advantage of the available bandwidth. Instead, we opt for a dynamic assignment of priorities based on Earliest Deadline First (EDF), whose applicability to CAN protocols has been widely studied (see *e.g.* [DN00]). Under the self-trigger implementation, deadlines for the control messages are given by (III.12), whose computational complexity is small since χ is a simple function of the state. Under the periodic paradigm the control loops always demand the same bandwidth. Our proposed scheme rather drops the periodicity assumption in favour of a dynamic allocation of bandwidth according to the state of the system. Transmission deadlines can still be guaranteed through an offline analysis, to be carried out in Section IV-C. In this paper we shall analyze the implementation over CAN, but extensions to other similar networks that include time-triggered slots (*e.g.* TTCAN) are discussed in Section VI.

B. Scheduling CAN under EDF

In order to take full advantage of the available resources, the different messages generated by the nodes on the network will be scheduled according to Earliest Deadline First (EDF), which is known to be optimal for non-idle schedules. Indeed, in the context of message scheduling for the CAN bus, EDF outperforms rate monotonic (RM) by 20% in network utilization, under realistic loads [ZS97]. EDF has been widely used in the context of real-time processors, while his applicability to CAN has been studied in [DN00] and [PA02], and successfully implemented in [FRB05].

Under the CAN protocol, each message contains an identifier field (29 bits wide for the extended format). This identifier is used for bus arbitration: the lower the identifier, the higher the priority. The standard requires that all messages must have a unique identifier. Under EDF, the priorities are assigned according to the deadlines: the shorter the deadline, the higher the priority and therefore the smaller the identifier. In run time such deadlines vary so the priorities need to be modified as well, incurring in a tolerable computational cost for each node (less than 5% of CPU time [DN00]). Since the range of deadlines might be very large (for instance, in the case of two control loops with very different time scales, *e.g.* temperature control and servo control), usually logarithmic scales are used to encode deadlines into priorities. Another possibility is to use several different time units (coarse time grain for far deadlines, fine time grain for shorter deadlines [RC98]). For more details on the implementation of EDF, we refer the interested reader to [DN00].

C. Offline schedulability analysis

Equation (III.12) describes the dynamic real-time requirements for control loops. Since offline the scheduler is oblivious to the real evolution of the plants, worst-case transmission times need to be considered to guarantee stability of the loops. As in the case of periodic transmission of control messages, enough bandwidth needs to be reserved offline to account for worst-case conditions, even though these might rarely occur. However, when the system starts running, the current state of the plant starts being measured, each node is aware of its real-time requirements (namely, transmission times for the messages), and reserved resources can be redistributed among all existing nodes. This property represents the main advantage of self-triggered control.

We characterize messages with the 3-tuple (T, C, d), where T represents the period of the message (or the minimum inter-arrival time for aperiodic messages), C the maximum transmission time, and d the relative deadline of the message (not necessarily equal to T). As mentioned before, each control loop involves two different types of messages: sensor messages (from sensor to controller) and control messages (from controller to actuator). Notice that sensor messages are followed by controller messages, hence they cannot be released at the same time. To model this precedence constraint, we consider a phase offset ϕ_c for the control message equal to the arrival time for the sensor message d_s plus the time C_s that the control node needs to evaluate the control law [ZS95] (since the control message cannot be released right after the arrival of the sensor message). The inter-arrival time for both messages $T = \tau_i$ is given by equation (III.12). The deadline d of the control message represents the delay between measurement and actuation, which was previously denoted by Δ_i. Performance of a control loop is determined by τ_i and Δ_i, that is, the inter-arrival time and the relative deadline of the control message. Since a lower delay usually entails a better performance for the control loop, it is important to find the minimum value for the deadlines that guarantees schedulability, as described below.

Given a set of n hard messages plus p control loops, the schedulability conditions (sufficient and necessary) under non-preemptive EDF are [ZS94]:

$$\sum_{i=1}^{n+2p} \frac{C_i}{T_i} \leq 1 \tag{IV.1}$$

$$\sum_{i=1}^{n+2p} \left\lceil \frac{t - d_i - \phi_i}{T_i} \right\rceil^{+} C_i + C_m \leq t, \quad \forall t \in S \tag{IV.2}$$

where the set S is defined as:

$$S = \cup_{i=1}^{n+2p} S_i,$$
$$S_i = \left\{ d_i + mT_i : m = 0, 1, \ldots, \left\lfloor \frac{t_{max} - d_i - \phi_i}{T_i} \right\rfloor \right\},$$

$$t_{max} = \max \Big\{ d_1, \ldots, d_{n+2p}, (C_m + \sum_{i=1}^{n+2p} (1 - d_i/T_i) C_i)/(1 - \sum_{i=1}^{n+2p} C_i/T_i) \Big\}$$

and $C_m := \max_i C_i$ is the maximum transmission time for all possible messages, $\lceil x \rceil^+ = \min\{n \in \mathbb{N}_0 | n \geq x\}$ and $\lfloor x \rfloor = \max\{n \in \mathbb{Z} | n \leq x\}$, and $\phi_s = 0$ for all sensor messages. The offset ϕ_c for control messages is equal to the deadline of the corresponding sensor message plus the computation time C_s. For a given set of control loops and a desired performance for each loop, we can derive the minimum inter-arrival time T for each control system, which corresponds to:

$$T = \min_{x_0} \{\tau(x_0)\}, \qquad x_0 \in \Omega$$

with τ defined according to equation (III.12), and Ω is the operating region for the control system. Notice that the offline schedulability analysis has to be based on the worst inter-arrival time for several reasons: 1) the initial condition x_0 of the system is unknown; 2) the plant model is inaccurate; and 3) there exist disturbances that might steer the system to such worst-case state. Nevertheless, the inherent dynamic structure of self-triggered control permits the online redistribution of the reserved resources.

There are also several related questions involving message schedulability that can be answered by relaxing equations (IV.1) and (IV.2), as explained herein below.

Q. *Given a schedule and a set of messages $\{M_i\}_{i=1,\ldots,n}$ is the system schedulable if p control systems (with desired performance σ) are added to the network? What is the maximum performance that those control loops can have while preserving schedulability?*

A. We consider n messages $\{M_1, \ldots, M_n\}$, each one defined by the tuple $M_i = (T_i, C_i, d_i)$. We add a set of sensor messages $\{M_{n+1}, \ldots, M_{n+p}\}$ and the corresponding control messages $\{M_{n+p+1}, \ldots, M_{n+2p}\}$, with their corresponding timing parameters. It is straightforward to evaluate conditions (IV.1) and (IV.2) to check the schedulability of the system. If the system is not schedulable for the desired quality-of-control, conditions (IV.1) and (IV.2) can still be satisfied by degrading the performance of the control loops. In that case, we would be interested in finding the maximum performance that we can have under the schedulability constraints. Should the system be schedulable, we would like to know how to increase the performance of the system. To answer both questions, it is easier to resort to the sufficient conditions developed in [ZS94], extended to account for phase offset. Without loss of generality, we assume:

$$d_1 + \phi_1 < d_2 + \phi_2 < \ldots < d_{n+2p} + \phi_{n+2p}$$

Then, the system is schedulable if the following inequalities are satisfied:

$$\textstyle\sum_{i=1}^{n+2p} \frac{C_i}{T_i} \leq 1 \qquad \text{(IV.3)}$$

$$\textstyle\sum_{i=1}^{k} \Big(1 + \frac{d_k + \phi_k - d_i - \phi_i}{T_i}\Big) C_i + C_m \leq d_k + \phi_k, \quad \forall k \in K \qquad \text{(IV.4)}$$

with $K = \{1, 2, \ldots, n\} - \{k : d_k + \phi_k = d_{k+1} + \phi_{k+1}, 1 \leq k \leq n + 2p - 1\}$. We have two decision variables for each control message, namely the minimum inter-arrival time and the associated relative deadlines. At a first stage we consider the deadlines to be equal to the periods $d_i = T_i$, except for the sensor and control messages, for which we set $d_i = T_i/2$ (since every T_i units of time a sensor and a control message need to be transmitted). Assigning different weights w_i to each control loop, the problem can be cast as a minimization of the cost function $J = \sum_i w_i \sigma_i$, under the constraints (IV.3) and (IV.4). Defining the auxiliary variables $f_i = 1/T_i$, it is straightforward to see that the problem can be posed as a linear programming problem, with f_i as the decision variables and performance σ_i as described by (III.12).

At the second stage, we can try to reduce the relative deadlines by making use of the available slackness once the control messages have been scheduled, according to the following problem.

Q. *Given a schedule for a set of sensor messages $\{M_1, \ldots, M_p\}$ and a set of control messages $\{M_{p+1}, \ldots, M_{2p}\}$, how much can we reduce the deadlines d_i while still guaranteeing schedulability?*

A. The sufficient condition (IV.3) is not affected by changes in the deadlines, so only equation (IV.4) needs to be analyzed. We propose two different ways to specify new deadlines:

- Let the new deadlines be: $\hat{d}_i = \alpha d_i, \forall i = 1, \ldots, 2p$, for some parameter $\alpha > 0$ that we seek to determine. While the offset for sensor messages is always zero, for control messages the offset ϕ_i changes as the deadline of the corresponding sensor message d_{i-p} is modified:

$$\hat{\phi}_i = \alpha d_{i-p} + C_s, \qquad i \geq p + 1$$

In that case equation (IV.4) turns into two inequalities:

$$\sum_{i=1}^{k} \Big(1 + \alpha \frac{d_k - d_i}{T_i}\Big) C_i + C_m \leq \alpha d_k, \forall k \in K, k \leq p \qquad \text{(IV.5)}$$

and

$$\sum_{i=1}^{p} \Big(1 + \alpha \frac{d_k - d_i}{T_i}\Big) C_i + \sum_{j=p+1}^{k} \Big(1 + \alpha \frac{d_k + d_{k-p} - d_j - d_{j-p}}{T_j}\Big) C_j + C_m \leq \alpha d_k + \alpha d_{k-p} + C_s, \forall k \in K, k \geq p + 1 \qquad \text{(IV.6)}$$

Solving inequalities (IV.5) and (IV.6) for α, we obtain a lower bound in the reduction of deadlines that preserve schedulability with the same periods T_i as before.

- Let the new deadlines be: $\hat{d}_i = d_i + \delta d_i$, with δd_i to be determined. In that case equation (IV.4) turns into the following inequality:

$$\textstyle\sum_{i=1}^{k}\left(1+\frac{(d_k+\delta d_k+\phi_k+\delta\phi_k)-(d_i+\delta d_i+\phi_i+\delta\phi_i)}{T_i}\right)C_i$$
$$+C_m \leq d_k + \delta d_k + \phi_k + \delta\phi_k \quad \forall k \in K \qquad \text{(IV.7)}$$

  We can now define different weights w_i for each deadline d_i, and then the problem can be solved again as a linear programing optimization problem of the form:

$$\text{minimize} \sum_{i=1}^{n} w_i \delta d_i \qquad \text{(IV.8)}$$
$$\text{subject to } A(\delta d) \leq b \qquad \text{(IV.9)}$$

  where the decision variables are δd, and A and b can be easily derived from (IV.7).

Once the problem of offline scheduling has been solved, we study how to dynamically allocate those resources that are released in run time by the control nodes.

D. Dynamic bandwidth allocation

Since the schedulability analysis was based on worst-case computations, spare bandwidth will be available in run time. In order to decide which message takes this extra bandwidth, different procedures can be followed. One simple solution would be to grant the available bandwidth to soft messages. The soft messages, having lower priorities than control messages or other hard messages, will only be sent through the network whenever the channel is idle.

A more versatile solution consists in assigning the spare bandwidth among all the existing nodes according to their needs. We propose a two-step process to solve the dynamic bandwidth assignment problem. At a high level we specify a set of functions that describe performance indices for each node, without considering the practical implementation details of the network. At a lower level we define a set of functions that deal with the concrete aspects of the implementation. These two different layers facilitate a separation of concerns: for instance, a control engineer only needs to define the high level functions, without taking into account the details of the implementation.

Since decisions need to be taken locally, we propose a persistent strategy where each node attempts to send messages continuously, but with dynamic priorities that vary according to the local status of each node. A particular case of such strategy was developed in [VFL$^+$04] and [WY01], where resources were granted to the control loop with the largest error. In our case, we have at our disposal the analytical formula (III.12) for the control loops, relating performance and resource requirements. If similar formulas are available for the other messages in the system, we can define a general performance index J that includes all messages in the network. For instance, given p control messages, n hard messages and m soft messages, we search for a decentralized policy that maximizes the following function:

$$J = \sum_{i=1}^{p} \bar{h}_i(\bar{\sigma}_i) + \sum_{i=1}^{n} \tilde{h}_i(\tilde{\sigma}_i) + \sum_{i=1}^{m} \hat{h}_i(\hat{\sigma}_i) \qquad \text{(IV.10)}$$

where $\bar{h}_i$, $\tilde{h}_i$ and $\hat{h}_i$ are functions defining the utility gained by sending a message, and $\bar{\sigma}_i$ $\tilde{\sigma}_i$, $\hat{\sigma}_i$ are functions relating local performance and bandwidth. Notice that for control nodes all the performance abstractions σ are in fact a function of the time between two consecutive transmissions τ_i (as implicitly defined by equation (III.12)), while for other nodes performance might depend on the number of transmissions. An example of a performance index for the case of a real-time video stream is shown in Figure 2. If $1/\sigma_R$ is the desired stream rate, there is no advantage in processing the video faster than required ($\sigma < \sigma_R$), since the user will barely notice any difference. At the same time it is not worthwhile processing frames at a very low rate since the quality of service at such rate is as unacceptable as not transmitting any message.

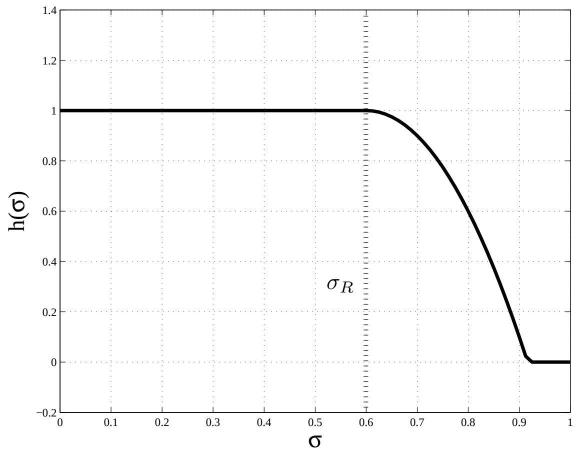

Fig. 2. Example of performance index $h(\sigma)$ for video streaming.

As the scheduler cannot be clairvoyant, our conjecture is that the best policy is to maximize the function J at each instant of time. Notice that information is only available locally, that is, each node is oblivious to the state of the other nodes, and thus it needs to make a decision only based on its current state. The term $h_i(\sigma_i)$ represents the improvement in performance if node i sends a message at time instant t. For simplicity, and without loss of generality, we assume that the improvement in performance is 0 whenever a message is not sent. Each node can thus evaluate its own performance $h(\sigma)$ and set accordingly the priority for the next message to be sent (the higher the performance, the higher the priority). Under the CAN protocol, only one message can be transmitted at a given time. Hence, at each instant of time all terms of the function (IV.10) are zero except for the node transmitting the message. The message with the largest $h_i(\sigma_i)$ at time t will maximize the overall performance index J and therefore should be the one with the highest priority. Since EDF is our scheduling policy, such message should have the shortest deadline.

In other words, under this persistent strategy every node attempts to deliver its messages continuously but with different priorities, according to the expected improvement in performance. For example, in the case of a temperature sensor, right after sending a message, the sensor will attempt to send another message again with the current value of the temperature. However, the benefit of such message is really low, since the temperature has barely changed since the last measurement. Therefore the priority of that message will be low and it will only be delivered if there is no message with higher priority trying to use the network. Nonetheless, as time evolves, the priority of the sensor message increases whenever the temperature varies. Another example is shown in detail in Figure 3, where 3 nodes compete for the channel. At the first time instant $k = 1$, each node evaluates its own performance index $h(\sigma)$ and sets accordingly its priority. The network is assigned to node A since $h_A > h_B > h_C$. Right after sending the message, the performance index of node A diminishes abruptly since sending another message would barely improve its performance. For $k = 2$, the same process is repeated and since $h_B > h_C > h_A$ the network is assigned to node B.

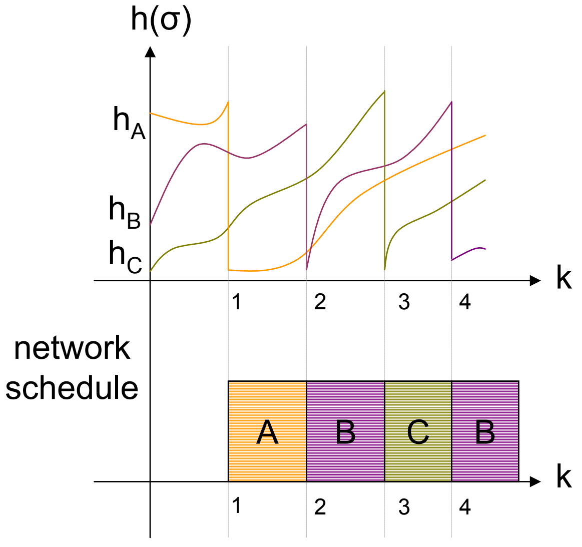

Fig. 3. Dynamic priorities for network scheduling.

In order to guarantee the hard deadlines under the aforementioned policy, constraints need to be imposed on the choice of the performance functions h. As mentioned before at the beginning of this subsection, we will follow a two-step approach. Instead of imposing such constraints on h, we shall rather define another function g at a lower level that computes the deadlines d_i from the performance function h but taking into account practical issues, such as finite number of bits for the priorities, unique assignment of priorities, etc. The function g would also be responsible for guaranteeing that no soft message inherits a higher priority than a hard message. Hence the deadlines are given by the following expression:

$$d_i = g_i(h_i(\sigma_i)), \qquad h \in \{\bar{h}, \tilde{h}, \hat{h}\} \tag{IV.11}$$

where $d_i \in \{0, 1, \ldots, 2^{Nb} - 1\}$ (set of all possible deadlines), with Nb the number of bits aimed at encoding the deadline. In this proposed two-step scheme, the function h defines the description of quality of service for each message while g plays a supervisory role guaranteeing fairness among the competing nodes. For that purpose, we reserve a range of deadlines [d_{min},d_{hard}] for hard messages (namely those that need to be delivered to guarantee the required performance) and another range [d_{soft},d_{max}] for soft messages (with $d_{\text{soft}} > d_{\text{hard}}$). It is easy to see that under EDF the soft messages will only be transmitted when no hard message needs to be sent. In this way, the dynamic strategy of bandwidth allocation preserves the schedulability of the hard messages, given by equations (IV.1) and (IV.2). Let $\breve{\sigma}_i$ be the minimum required performance (*i.e.* , σ_i is allowed to vary between 0 and $\breve{\sigma}_i$), and $\breve{d}_i$ the corresponding deadline. While there are numerous ways to define the set of functions g and h, there are several properties that need to be satisfied so that that all hard deadlines are met:

- Property 1: a higher improvement implies a shorter deadline:

$$h_i(\sigma_i) > h_l(\sigma_l) \Rightarrow g_l(h_l(\sigma_l)) > g_i(h_i(\sigma_i))$$
$$\forall \sigma_i \in [0, \breve{\sigma}_i[, \qquad \sigma_l \in [0, \breve{\sigma}_l[\qquad i, l \in \{1, 2, \ldots, p\} \tag{IV.12}$$

  In particular, for $i = l$, equation (IV.12) implies that each function g_i is monotonic decreasing in its argument $h(\sigma)$. This property suggests as well that it is not necessary to assign deadlines according to h (which might have a cumbersome expression), but instead it is enough to deal with a simplified version g that renders the same assignment of priorities.
- Property 2. All messages that need to be sent in order to guarantee the required performance $\breve{\sigma}$ will have higher priority than those messages leading to an improvement of performance above the required performance (regardless of the actual value of h).

$$g_i(h_i(\breve{\sigma}_i)) < g_l(h_l(\sigma_l))$$
$$\forall \sigma_l \in [0, \breve{\sigma}_l[, i, l \in \{1, 2, \ldots, p\} \tag{IV.13}$$

  Together with property 1, this specifies that the hard messages will have deadlines in the range [d_{min},d_{hard}], while the soft messages will have deadlines in the range [d_{soft},d_{max}]. All extra messages that were not considered in the offline analysis are only delivered when no hard message is competing for the channel and thus schedulability is guaranteed.
- Property 3. In the offline analysis, a minimum inter-arrival time T was considered for each message. To preserve schedulability, no deadline smaller than $\breve{d}_i = T$ should be used for the EDF policy.

$$\forall \sigma_i \in \mathbb{R}^+ \qquad g_i(h_i(\sigma_i)) \geq \breve{d}_i \tag{IV.14}$$

  This property guarantees a minimum bandwidth for each node and implicitly provides isolation between nodes. In this way greedy nodes are not allowed to overload the network, preventing the use of deadlines shorter than the prescheduled; such deadlines could violate the schedulability conditions (IV.1) and (IV.2) in worst case operation. Moreover, the offline schedulability analysis shows that setting deadlines $\check{d}_i$ is enough to guarantee the required performance.

The set of functions g_i should also guarantee a unique assignment of deadlines, that is, there are not two messages with the same priority at the same time, which would produce a collision in the network (this can be achieved by allocating several bits of the identifier field to identify the sending node). Should the set of functions g_i satisfy these properties, the performance of the control loops is always at least equal to the required performance, i.e., $\sigma_i \leq \check{\sigma}_i$, since all the pre-scheduled deadlines $\check{d}_i$ are always satisfied. This can be summarized in the following proposition.

Proposition 4.1: Consider a set of n hard messages, with a given required performance, which is schedulable under the EDF policy. If deadlines are assigned according to (IV.11), with g satisfying properties (IV.12), (IV.13), and (IV.14), all nodes will attain at least the required performance.
Moreover, the proposed strategy aims at maximizing the performance index (IV.10) at each time step.

V. Example

To show the benefits of the proposed approach, we consider a train car equipped with a CAN shared by 4 control loops and other several nodes delivering hard and soft messages. The control loops correspond to the control of four wheels with magnetic suspension. The hard messages to be delivered through the network comprise the current in the power line cable, that needs to be monitored every 4 msec, and the internal temperature of the coils, that is monitored every 10 msec. The soft messages mainly consist in monitoring auxiliary sensors for fault detection, plus several multimedia streams for the passengers. For the magnetic suspension system, we borrow the model proposed in [BC96]:

$$\begin{aligned}\dot{\mathbf{x}}_1 &= \mathbf{x}_2\\ \dot{\mathbf{x}}_2 &= g_c - \frac{C}{m}\left(\frac{\mathbf{x}_3}{\mathbf{x}_1}\right)^2\\ \dot{\mathbf{x}}_3 &= -\frac{R}{L}\mathbf{x}_3 + \frac{2C}{L}\left(\frac{\mathbf{x}_2\mathbf{x}_3}{\mathbf{x}_1^2}\right) + \frac{1}{L}\mathbf{u}\end{aligned} \qquad \text{(V.1)}$$

where $\mathbf{x}_1$ represents the wheel vertical position, $\mathbf{x}_2$ is the wheel vertical velocity and $\mathbf{x}_3$ is the applied voltage in the magnet. The term L represents the coil's inductance, R the coil's resistance, g_c the gravitational constant, C the magnetic force constant and m is the mass of the wheel. A nonlinear feedback law of the form $u = \rho(x_1, x_2, x_3)$ is designed to steer the system to the desired vertical position. Following the steps in [AT08b], we derive a self-trigger condition relating performance and transmission times given a desired position x_{1d}:

$$\tau(\sigma) = \frac{-1.11 + \arctan(2 + 3\sigma)}{1 + z_1^2 + (z_1 + z_2)^2 + (z_1 + z_2 + z_3)^2} \qquad \text{(V.2)}$$

σ	periodic	self-triggered
0.3	0.864	0.083
0.4	0.752	0.072
0.5	0.704	0.067

TABLE I
UTILIZATION FACTOR OF THE CONTROL LOOPS UNDER THE PERIODIC AND THE SELF-TRIGGERED IMPLEMENTATION FOR A SIMULATION OF 3S.

where:

$$\begin{aligned} z_1 &= x_1 - x_{1d}\\ z_2 &= x_2\\ z_3 &= g_c - \frac{C}{m}\left(\frac{x_3}{x_1}\right)^2\end{aligned} \qquad \text{(V.3)}$$

We consider a 1Mbit/s CAN bus. For simplicity, we assume all transmission times to be equal, $C = 130\mu$s. Equation (V.2) gives us a minimum inter-transmission time T_s for the control messages of 1.47ms for a quality of control $\sigma = 0.5$ (we consider the same value of σ for each wheel). The computation time for the control law C_s is considered to be less than C. By checking the schedulability conditions (IV.1) and (IV.2) we conclude that the system is schedulable for the desired quality-of-control $\sigma = 0.5$ of each control loop.

Next we compare the behaviour of the control loops under a periodic and a self-trigger implementation, for a fixed value of $\sigma = 0.5$, and a desired position $x_{1d} = 5mm$, assuming no dynamic bandwidth allocation in the network. The system is simulated for 3 seconds, which corresponds to the duration of the transient response for the control system. The evolution of the position of one of the wheels is displayed in Figure 4. While the periodic implementation always sends messages at the same rate ($1/T_s$), the self-triggered implementation enlarges the inter-transmission times as the state reaches the desired position x_{1d}. Table I shows the bandwidth utilization for both implementations, for different quality-of-control requirements. It is clear from Table I and Figure 4 that the self-trigger mechanism provides a quality of control similar to a periodic implementation while reducing the consumed bandwidth by 90%.

Regarding the dynamic bandwidth allocation, it is necessary to define the performance indices h for each message in the network. Messages coming from the current and temperature sensors will have their priorities increased whenever the corresponding magnitudes differ from their nominal values:

$$\begin{aligned} h_{\text{current}} &= (i - i_{ref})^2\\ h_{\text{temp}} &= (t - t_{ref})^2\end{aligned}$$

where i represents the current, t the temperature and the nominal values are $i_{ref} = 0.1$A and $t_{ref} = 30°$C. For the multimedia messages, we define the corresponding functions h according to Figure 2. For ease of explanation, the control messages will not be part of the global performance function, hence only messages leading to the required performance $\sigma = 0.5$ will be delivered by the sensor and control nodes. We consider standard profiles for the current and the temperature,

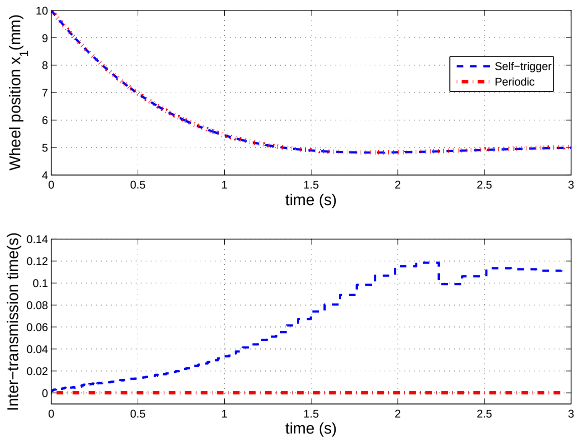

Fig. 4. Wheel position and transmission times under the periodic and the self-triggered implementation.

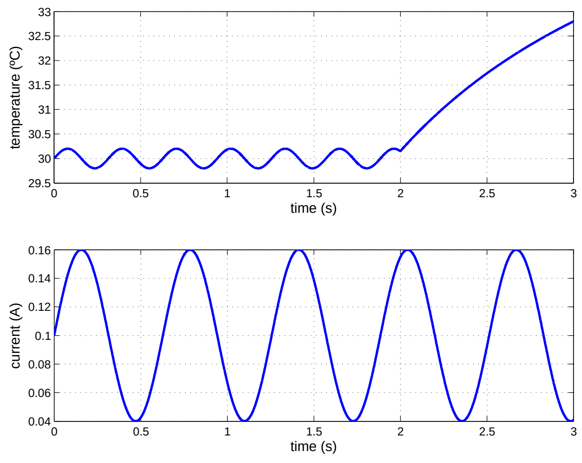

Fig. 5. Current and temperature profiles.

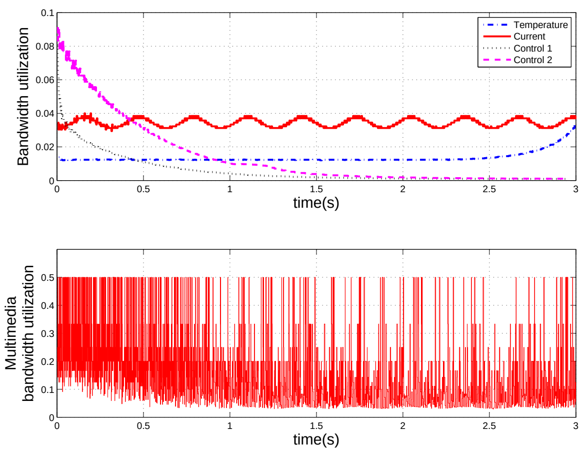

Fig. 6. Evolution of the utilization factor for the different nodes.

displayed in Figure 5. The results of the dynamic strategy are shown in Figure 6 where we can find the utilization factor (transmission time C divided by the inter-arrival time) for two of the control loops, for the two sensors, and for the multimedia stream. All nodes adjust their demands based on their present states. In the simulations different initial conditions were considered for the control loops, leading to a different evolution of the bandwidth utilization factor (since the transmission times are a function of the state). In all cases the required bandwidth is reduced as the control systems reach the reference point. The utilization factor for the current messages change according to the current in the coil. The longest inter-transmission times correspond to the points where the current is equal to the reference value. Likewise, the utilization factor for the temperature sensor raises whenever the temperature deviates from the reference point, since it leads to a higher improvement according to the definition of h_{temp}. Notice how the utilization factor for both sensors become larger than the preassigned bandwidth (0.033 for the current and 0.013 for the temperature) without violating the schedulability of the system, since they only borrow the bandwidth released by the control loops. Finally, the inter-transmission times for the multimedia messages oscillate widely since all these messages are soft messages and thus will be blocked by any hard message. In summary, the spare bandwidth is fairly distributed in run time among the different nodes based on their local states, while schedulability of the hard messages is preserved.

VI. Extension for TTCAN

During the 1990s, a new layer of the CAN protocol was developed to include time-triggered slots. Under this solution, named Time-Triggered CAN (TTCAN, see [FMD$^+$00]), access to the medium is determined by a predefined schedule. Three different types of windows are possible in the TTCAN schedule: exclusive, arbitrary and spare. Exclusive windows are preassigned to a unique node, and no other node can make use of the slot even if no transmission is carried by the preassigned node. Such slots provide a good isolation between nodes but might imply a waste of resources when no message is transmitted. On the other hand, in arbitrary windows any node can compete for the access to the bus under the priority-based mechanism used in standard CAN. Free windows are aimed at possible future expansions of the system. Under the TTCAN protocol, all control-related messages are usually assigned to exclusive windows to guarantee a minimum time between updates under all possible conditions. When worst-case conditions do not occur, those slots cannot be assigned to other nodes in need of higher bandwidth, because of the inherent rigid structure of the exclusive windows. We propose to place the messages on the arbitrary windows, so that preassigned bandwidth can be released whenever it is not needed. We can thus apply the same strategy as before but just in the arbitrary windows. Transmission deadlines are still guaranteed according to Proposition 4.1. But how to decide which slot is arbitrary or exclusive? This assignment needs to be known in advance by each node, in order to decide when to send messages through the network. Such information is encoded by means of a matrix schedule. The offline scheduling problem consists in the definition of such matrix. Systematic procedures to construct the matrix appeared in [NH07] and in [SS07].

To construct the matrix schedule, all control tasks need to be treated as operating in worst-case conditions. That is, they will be scheduled as periodic with the minimum inter-execution time provided by equation (III.12). Once the matrix schedule is defined, all exclusive windows that were alloted to control nodes will be redefined as arbitrary windows, to allow other tasks to send messages during those instants in time. For such policy, the following question arises:

Q. *Under such strategy, will control messages still meet their deadlines even when arbitrary windows are used instead of exclusive windows?*
A. It is straightforward to find a schedule meeting deadlines, namely one that assigns each control message to the previously assigned windows in the original matrix schedule. Since EDF is optimal for non-idle schedules, EDF finds a schedule meeting the deadlines.

VII. Conclusions

In this paper we have shown the benefits of self-trigger control in the context of networked control systems. We proposed a dynamic bandwidth allocation strategy that assigns resources to the nodes that are more in need. The results were illustrated in detail with the example of a train car equipped with a CAN, shared by several control loops. Extensions of the results to TTCAN protocols were also described. Due to the lack of space, we did not consider robustness of the self-trigger technique in this paper. We refer the interested reader to [MJT09] for a discussion of the topic.

Notice that the proposed approach can be used to analyze the effect of delays and packet dropouts on networked control systems, since the self-trigger conditions determine the maximum time that can elapse between two consecutive updates of the input value. For instance, a packet dropout will not cause instability of a control loop provided that the next message can be delivered before the self-trigger condition is violated.

VIII. Acknowledgements

The authors would like to thank Soheil Samii for helpful discussions on the topics of this paper. This research was partially supported by the National Science Foundation awards 0712502, 0834771 and 0841216 and Mutua Madrileña Automovilista.

References

[AHBLP06] A.T. Al-Hammouri, M.S. Branicky, V. Liberatore, and S.M. Phillips. Decentralized and dynamic bandwidth allocation in networked control systems. In *14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2006.

[AT08a] A. Anta and P. Tabuada. On the benefits of relaxing the periodicity assumption for control tasks. *RTAS Work-in-progress track.*, 2008.

[AT08b] A. Anta and P. Tabuada. To sample or not to sample: Self-triggered control for nonlinear systems. *Accepted for publication. arXiv:0806.0709*, 2008.

[Bam03] B. Bamieh. Intersample and finite wordlength effects in sampled-data problems. *Automatic Control, IEEE Transactions on*, 48(4):639–643, 2003.

[BC96] W. Barie and J. Chiasson. Linear and nonlinear state-space controllers for magnetic levitation. *International Journal of Systems Science*, 27(11):1153–1163, 1996.

[CHL+03] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.E. Arzen. How does control timing affect performance? *Control Systems Magazine, IEEE*, 23(3):16–30, 2003.

[DN00] M. Di Natale. Scheduling the CAN bus with earliest deadline techniques. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pages 259–268, 2000.

[FMD+00] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, M. Walther, and R.B. GmbH. Time Triggered Communication on CAN (Time Triggered CAN-TTCAN). In *Proceedings of International CAN Conference, Amsterdam, The Netherlands*, 2000.

[FRB05] S. Fuster, F. Rodriguez, and A. Bonastre. Software-Based EDF Message Scheduling on CAN Networks. In *Embedded Software and Systems, 2005. Second International Conference on*, pages 450–455, 2005.

[Gmb91] Robert Bosch GmbH. CAN Specification Version 2.0. 1991.

[Kha02] H.K. Khalil. *Nonlinear systems*. Prentice Hall Upper Saddle River, NJ, 2002.

[MJT08] M. Mazo Jr and P. Tabuada. On Event-Triggered and Self-Triggered Control over Sensor/Actuator Networks. *47th IEEE Conference on Decision and Control, Cancún, Mexico*, 2008.

[MJT09] M. Mazo Jr and P. Tabuada. On the robustness of self-triggered control for sensor/actuator networks. *48th IEEE Conference on Decision and Control*, 2009.

[NH07] M. Naughton and D. Heffernan. A new real-time message scheduling tool for control networks. *Industrial Robot: An International Journal*, 34(3):188–194, 2007.

[NTC09] D. Nesic, A.R. Teel, and D. Carnevale. Explicit computation of the sampling period in emulation of controllers for nonlinear sampled-data systems. *IEEE Transactions on Automatic Control*, 54(3):619–624, 2009.

[PA02] P. Pedreiras and L. Almeida. EDF message scheduling on controller area network. *Computing & Control Engineering Journal*, 13(4):163–170, 2002.

[RC98] F. Rodríguez and J.C. Campelo. EDF message scheduling on a CAN network. In *5th International CAN Conference*, 1998.

[SLSS96] D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin. On task schedulability in real-time control systems. *17th IEEE RTSS*, 1996.

[SS07] K. Schmidt and E.G. Schmidt. Systematic Message Schedule Construction for Time-Triggered CAN. *Vehicular Technology, IEEE Transactions on*, 56(6 Part 1):3431–3441, 2007.

[Tab07] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[VFL+04] M. Velasco, J.M. Fuertes, C. Lin, P. Marti, and S. Brandt. A control approach to bandwidth management in networked control systems. In *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, volume 3, 2004.

[WL09] X. Wang and M. Lemmon. Self-triggered feedback control systems with finite-gain L 2 stability. *IEEE Transactions on Automatic Control*, 45(3):452–467, 2009.

[WY01] G.C. Walsh and H. Ye. Scheduling of networked control systems. *Control Systems Magazine, IEEE*, 21(1):57–65, 2001.

[ZS94] Q. Zheng and K.G. Shin. On the ability of establishing real-time channels in point-to-pointpacket-switched networks. *Communications, IEEE Transactions on*, 42(234):1096–1105, 1994.

[ZS95] K.M. Zuberi and K.G. Shin. Non-Preemptive Scheduling of Messages on controller Area Network for Real-Time Control Applications. *IEEE Transactions On Robotics And Automation*, 1995.

[ZS97] K.M. Zuberi and K.G. Shin. Scheduling Messages on Controller Area Network for Real-Time CIM Applications. *IEEE Trans. Robotics and Automation*, 13(2):310–314, 1997.