

On Event-Triggered and Self-Triggered Control over Sensor/Actuator Networks

Manuel Mazo Jr. and Paulo Tabuada

Abstract—Event-triggered and self-triggered control have been recently proposed as an alternative to the more traditional periodic execution of control tasks. The possibility of reducing the number of executions while guaranteeing desired levels of performance makes event-triggered and self-triggered control very appealing in the context of sensor/actuator networks. In this setting, reducing the number of times that a feedback control law is executed implies a reduction in transmissions and thus a reduction in energy expenditures. In this paper we introduce two novel distributed implementations of event-triggered and self-triggered policies over sensor/actuator networks and discuss their performance in terms of energy expenditure.

I. INTRODUCTION

Sensor networks research has extensively dealt with the extraction of information from the physical world. Many of the developed applications concentrate on how to obtain this information for posterior off-line analysis [1], [2] or on-line processing of this information for tracking [3], [4], distributed optimization [5] or mapping [6]. In all of these applications there is a common desire for small power consumption, which would extend the life span of the network. Many of the approaches used to reduce the power consumption concentrate on the communication requirements, as the works on compressive sensing [7], network throughput optimization [8], [9], message-passing algorithms [10] or sleep-scheduling of the nodes [11]. Still, most of these studies are performed only for on- or off-line analysis on the information gathered.

We address the problem of minimizing energy consumption for applications in which actuation plays a major role, namely control applications. In this context, energy expenditures can be directly related to the frequency at which measurements are being taken and transmitted through the network. Although the choice of the sampling frequency is a problem as old as the use of microprocessors for control, this question never received a definitive answer and engineers still rely on rules of thumb [12], [13], [14] such as sampling with a frequency 20 times the system bandwidth. Moreover, the choice of *periodic* sampling is not even a natural one since the behavior of a control system can be quite different in different regions of the state space.

In this paper, we show how to minimize energy consumption by resorting to *event-triggered* and *self-triggered*

sampling strategies over sensor/actuator networks (SAN). The event-triggered idea has been previously explored in the literature [15]. It has been shown in [16] that by sampling and computing the controller *only when* a certain threshold condition on the state is violated, one can sample less frequently than with a periodic scheme while guaranteeing the same performance. In [17] and [18] the self-triggered sampling paradigm was explored, for linear and non-linear systems respectively, in an attempt to eliminate the requirement of continuous state measurement. Self-triggered strategies compute the next sampling time using the last state measurement and thus do not require knowledge of the current state. The contribution of this paper is to show how the techniques introduced in [16], and later refined in [19], [17] for H_∞ controllers, can be implemented over sensor-actuator networks to considerably reduce the number of network transmissions. We propose an event-triggered strategy in which each node uses its local information to determine when to make a transmission and a self-triggered strategy in which the actuator node determines for how long should the sensing nodes sleep before collecting and transmitting fresh measurements.

The paper is organized in the following way: Section II introduces the notation and states the problem. Then we proceed to describe the proposed decentralized self-triggered algorithm in Section III. Next we propose a distributed event-triggered protocol in Section IV, followed by Section V where we formalize how to obtain the times between updates in the previously described algorithms. The proposed techniques are illustrated with two examples in Section VI. The paper concludes with a brief discussion in Section VII.

II. NOTATION AND PROBLEM STATEMENT

We consider a linear control system:

$$\dot{x} = Ax + Bu, \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m \quad (1)$$

stabilized by a linear controller:

$$u = Kx \quad (2)$$

In what follows, we assume that K was obtained through a LQR design and consider its implementation over a sensor/actuator network. Therefore, K is given by $K = -B^T P$ where P is the solution of the matrix Riccati equation:

$$A^T P + PA - Q + L = 0 \quad (3)$$

with $Q = PBB^T P$ and for some $L \geq 0$ and $P = P^T > 0$. Furthermore, we assume $m=1$, for simplicity of presentation, and also the existence of a distinguished node collocated

This work has been partially funded by the Spanish Ministry of Science and Education/UCLA fellowship LA2004-0003 and by the NSF CSR-EHS 0712502 grant.

M. Mazo Jr. and P. Tabuada are with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095-1594, USA mmazo,tabuada@ee.ucla.edu.

with the actuator called the actuator node. The remaining nodes are sensor nodes and further described below. The control signal is obtained from zero-order sample and hold measurements so that:

$$u(t) = Kx(t_k), \quad t \in [t_k, t_{k+1})$$

where t_k and t_{k+1} are two consecutive sampling instants. For the sake of readability, we drop the explicit dependence of t but denote with the subindex k those quantities that are constant and equal to the value at the update time t_k , i.e. $x_k = x(t_k)$. With this kind of sampled feedback the closed loop is described by:

$$\dot{x} = Ax + BKx_k = (A + BK)x + BKe$$

where e represents a measurement error defined by:

$$t \in [t_k, t_{k+1}) \implies e(t) = x(t_k) - x(t)$$

Treating the measurement error e as a new variable we can rewrite the closed loop dynamics as:

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A + BK & BK \\ -A - BK & -BK \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}$$

since $\dot{e} = -\dot{x}$. Moreover, the matrix P , obtained as the solution of (3), defines a quadratic Lyapunov function $V = x^T Px$ whose derivative along (4) satisfies:

$$\dot{V} = -x^T Lx + e^T Qe - u_k^T u_k$$

The desired performance for the closed loop system can be described by a matrix $S > 0$ for which the inequality:

$$\dot{V} \leq -x^T Sx \quad (4)$$

holds. The matrix S would then define the desired rate of decay for V . In order to enforce (4) we need to select the sampling instants t_k so that $-x^T Lx + e^T Qe - u_k^T u_k \leq -x^T Sx$. This can be achieved by sampling when (5) holds.

$$-x^T(L - S)x + e^T Qe - u_k^T u_k = 0 \quad (5)$$

At $t = t_k$ the execution of the controller renders $x_k = x$ and $e = x_k - x = 0$, forcing $-x_k^T(Q + L - S)x_k \leq 0$. Now we can appreciate how $(Q + L - S)$ captures the trade-off between sampling and performance. In a typical design one would pick L to design the continuous optimal controller, Q will be determined by the LQR design, and we will get to pick again S . The closer S and L are, the better the performance of the system will be (closer to the prescribed continuous design) at the expense of more frequent sampling.

In a practical setting the measurements x_k will actually be received after a certain delay Δ which should be incorporated into the definition of e . For the sake of simplicity we ignore this delay by assuming $\Delta = 0$ although we can incorporate the delay Δ in the proposed framework by appropriately selecting S in the rule (4) by following the approach described in [16].

We are interested in the use of event-triggered and self-triggered control techniques as they require fewer measurements than periodic techniques. In particular, in the context

of SAN's, this also means that less transmissions are required for control and thus less energy usage. We will assume without loss of generality that each sensor measures one entry of the full state of the system. With all of the above in mind, the problem we solve in this paper is:

Problem 2.1: Given a Linear Time Invariant system as in (1), a controller (2) rendering the closed-loop system asymptotically stable, design a distributed algorithm enforcing the controller update rule (4) over a SAN.

We can now see that the fundamental difficulty when implementing an event-triggered or self-triggered strategy in a distributed-sensing setting is that the ‘‘triggering’’ condition (4) has to be evaluated continuously and it depends on full state information. A continuous centralization of local measurements to check (4) would lead to extensive communication and unacceptably high energy consumption, thus justifying a distributed design. In addition to a reduction in energy consumption, we will also require the distributed algorithms to be energy-balanced, i.e. all the sensing nodes should consume energy at the same pace, since as soon as one sensing node exhausts its energy reserves the stability and performance of the closed loop system becomes seriously compromised.

In what follows $\|\cdot\|$ will denote the usual Euclidean norm of a vector and $\sigma_{m,M}(\cdot)$ the minimum (m) or maximum (M) singular value of a given matrix.

III. A SELF-TRIGGERED PROTOCOL

The first distributed implementation follows the self-triggered paradigm for control [20], [17], [18]. According to this paradigm, in addition to use the current state to compute the controller, the current state is also used to compute the next execution time of the controller. In this section we describe a particular protocol to implement this paradigm over a sensor/actuator network.

A. Algorithm description

We start by assuming the availability of a routing tree with root at the actuation node. The construction of a routing tree is typically done during the discovery phase of the network by resorting to several efficient algorithms available in the literature [21]. We also assume that the actuator node has access to larger energy reserves since it drives one or several actuators that are typically more power consuming. Therefore, we assume that the actuation node has the ability to broadcast messages to all nodes without seriously reducing its energy reserves.

The proposed decentralized algorithm for self-triggered control consists of two phases:

- A measurement collection phase in which the network computes in a distributed manner, as explained later in this section, two scalars: $u_k = Kx_k$ and $\|x_k\|^2$. The actuator node uses this information to update the actuator and to compute the next time instant at which new samples should be collected and transmitted.
- A broadcast phase during which the actuator node broadcasts the next update time to all the sensing nodes.

In practice, in order to avoid synchronization issues one would rather send $t_{k+1} - t_k$ instead of t_{k+1} . The sensing nodes would then sleep for $t_{k+1} - t_k$ units of time before starting the next measurement collection phase. These $t_{k+1} - t_k$ times will be computed in Section V.

In any implementation of a control system the control signal u has to be delivered to the controller. In addition to u_k we will also need to transmit additional information in order to predict the sampling times. For the techniques that we will present $\|x_k\|^2$ is enough; nevertheless, more information will help to increase the time between updates.

The distributed computation of $u_k = Kx_k$ and $\|x_k\|^2 = \sum_1^n x_i^2(t_k)$ is based on the fact that both quantities are sums of quantities locally available to the nodes of the network. The computation of these quantities can then be done by a *wave algorithm* for trees (*Tree Wave-Algorithm*) [21] in which each node collects its local quantity ($K_i x_k^i$ and $(x_k^i)^2$ respectively), adds it to the quantities that it receives from its children and transmits the resulting sum to its parent. Having the root at the actuation node ensures that the computation ends at the actuation node. The algorithm also ensures that each node transmits packets of the same size. The number of transmissions are also limited to one per node, due to the *Tree Wave-Algorithm* implementation. Briefly, the *Tree Wave-Algorithm* limits the communication by passing to the parent node a “token” or “signal” only when the node has received one from each of the links to its children, and it has locally generated its own token. In this way when the root receives a token, the root node knows that all nodes in the network generated locally a token.

If the maximum delay introduced by the measuring collection phase can be computed, it can be regarded as a computation delay and accommodated by modifying the triggering rule as in [16].

B. Energy utilization

In order to determine the energy consumed by the proposed protocol, we need to take into account the power consumed in transmitting data, in listening for transmissions, and in computing.

In the case of a self-triggered strategy the nodes only perform the computations needed in the wave algorithm computing u_k and $\|x_k\|^2$. We will denote this computing cost by D_1 . It is a fixed unit of energy consumed per update. The radio module can also be kept asleep most of the time as there is no need for communication between updates, therefore the listening cost could be neglected. As for the power consumed in transmitting information, the nodes on the self-triggered protocol only send two scalars per update. Thus, we represent this cost by two parameters: C_1 and C_2 . The first parameter describes the cost associated to the packet overhead transmission (independent of the payload) while the second parameter describes the cost per transmitted scalar. Therefore, in the self-triggered case there would be an energy cost of $C_1 + 2C_2$, and the total energy consumed at a node

at any given point in time will be given by:

$$J_1(t) = \sum_{k \in \Gamma(t)} (C_1 + 2C_2 + D_1) \quad (6)$$

where $\Gamma(t) = \{k : t_k \leq t\}$.

IV. AN EVENT-TRIGGERED PROTOCOL

A different strategy consists in checking in a distributed manner the condition imposed on the decay of the Lyapunov equation. This leads to a distributed computation of the controller update times. Our goal is to reduce the consumption at the nodes of the network, on which communication has the biggest impact. Therefore we cannot exchange information frequently between nodes. Instead we propose to have the nodes estimate in a conservative way when the triggering condition stops to hold. Each of the nodes will be making use of its local measurements in order to perform this estimation. When all the nodes agree in that the triggering condition has been violated a new controller update is forced.

A. Algorithm description

Once more we will need to assume the existence of a routing tree with root at the actuation node visiting all the nodes in the network. The event-triggered algorithm can be divided in several phases:

- Phase 1: The actuation node broadcasts a request for new measurements.
- Phase 2: The sensing nodes take their local measurement and compute their part of u_k and $\|x_k\|^2$ and forward them up the routing tree (towards the root).
- Phase 3: The actuation node finishes the computation of u_k and $\|x_k\|^2$ and broadcasts those quantities to the rest of the nodes.
- Phase 4: The sensing nodes use u_k and $\|x_k\|^2$ to update their local estimators accordingly. When a node’s local triggering condition stops to hold it will generate a local “token”, starting the computation of the *Tree Wave-Algorithm*, explained in a previous section. When the actuation node receives “tokens” from all of its children nodes it will broadcast a request for new measurements and the network would be back in Phase 1. The local triggering (generation of these tokens) will be explained in detail in Section V.

As the measurements from the nodes are synchronized, any delay introduced in phases 2 and 3 can be regarded as a computation delay and again dealt with modifying the triggering rule as in [16].

Note also that the performance of neither the self-triggered nor the event-triggered protocol is affected by the topology of the network if no delays are taken into account. However, the topology will have an effect on the maximum delay present in a practical implementation, and thus on its performance.

B. Energy utilization

The event-triggered protocol requires the nodes to continuously compute estimates, and listen to possible communications from their children. Therefore, we have a constant

power consumption due to listening and computation, which we will denote by l and d_2 respectively. In the event-triggered protocol there are two transmissions per node per update. The first one is used just to decide the next update time and contains no payload, therefore incurring on a cost of C_1 . The second transmission is used to compute the control signal u_k and $\|x_k\|^2$, therefore having two scalars as payload and incurring on a cost of $C_1 + 2C_2$. Finally the total energy consumption at any node at any point in time will be given by the following equation:

$$J_2(t) = \sum_{k \in \Gamma(t)} (2C_1 + 2C_2) + (l + d_2)t \quad (7)$$

where once more $\Gamma(t) = \{k : t_k \leq t\}$.

V. HOW TO COMPUTE SLEEPING TIMES

We will need to relax condition (4) in order to make it possible to check in a decentralized way. In this section we will present these relaxations and how to take advantage of them to perform a distributed or decentralized computation of the times between updates of the controller.

A. Self-triggered control

One of these possible relaxations provides the triggering condition:

$$e^T(Q + L - S)e \leq \frac{1}{2}\sigma_m(L - S)\|x_k\|^2 + u_k^T u_k \quad (8)$$

which implies the inequalities:

$$\begin{aligned} e^T(Q + L - S)e &\leq \frac{1}{2}x_k^T(L - S)x_k + u_k^T u_k \Rightarrow \\ \dot{V} &\leq -x^T Sx \end{aligned}$$

also hold, where we used:

$$x^T(L - S)x \geq \frac{1}{2}x_k^T(L - S)x_k - e^T(L - S)e$$

The right-hand-side of equation (8) is a linear combination of the constants u_k and $\|x_k\|^2$ obtained from a distributed computation involving only the last measurements taken by the sensing nodes. In order to check the triggering condition on-line we will need to estimate an upper bound for the evolution of $e^T(Q + L - S)e$.

Provided that L and S where designed so that $(Q + L - S) > 0$ we can find a dynamic estimator ϵ satisfying $\|e(t)\|_M \leq \epsilon(t)$ and bounding above $\|e\|_M = \|M^{\frac{1}{2}}e\| = \|\sqrt{Q + L - S}e\|$:

$$\frac{d}{dt}\|e\|_M \leq \|\dot{e}\|_M \leq \|M^{\frac{1}{2}}AM^{-\frac{1}{2}}\|\|e\|_M + \|(A + BK)x_k\|_M$$

Which now can be modified to be used as a bound estimator depending only on quantities computable over the network:

$$\frac{d}{dt}\|e\|_M \leq \alpha\|e\|_M + \beta(u_k, \|x_k\|) \quad (9)$$

with

$$\alpha = \|M^{\frac{1}{2}}AM^{-\frac{1}{2}}\| \quad (10)$$

$$\beta = \min\{\beta_1, \beta_2\} \quad (11)$$

$$\beta_1 = \|M^{\frac{1}{2}}(A + BK)\|\|x_k\| \quad (12)$$

$$\beta_2 = \left(\|M^{\frac{1}{2}}(A + BK)\|^2\|x_k\|^2 + \|M^{\frac{1}{2}}Bu_k\|^2 + 2\|u_k^T B^T M^{\frac{1}{2}}A\|\|x_k\| \right)^{\frac{1}{2}} \quad (13)$$

Our bound estimator would therefore be $\dot{\epsilon} = \alpha\epsilon + \beta$, which by the *Comparison Lemma* [22], would provide with $\epsilon(t) \geq \|e(t)\|_M$. Integrating ϵ and taking into account $\epsilon(t_k) = 0$ leads to:

$$\epsilon(t) = \frac{\beta}{\alpha} \left(e^{\alpha(t-t_k)} - 1 \right) \quad (14)$$

Therefore the next update time will be given by:

$$\epsilon^2(t_{k+1}) = \frac{\sigma_m(L - S)}{2}\|x_k\|^2 + u_k^T u_k \quad (15)$$

which can be solved analytically:

$$t_{k+1} = t_k + \frac{1}{\alpha} \log \left(1 + \frac{\alpha\kappa_k}{\beta} \right) \quad (16)$$

$$\kappa_k = \sqrt{\frac{\sigma_m(L - S)}{2}\|x_k\|^2 + u_k^T u_k} \quad (17)$$

Let us summarize these results in the following proposition:

Proposition 5.1: The distributed implementation of a LQR controller (2), designed to stabilize the linear system (1), defined by the algorithm described in Section III with update times given by (16) guarantees stability of the closed loop system and enforces $\dot{V} \leq -x^T Sx$. Moreover, the algorithm is energy-balanced.

B. Event-triggered control

A different alternative consists of designing estimators in which local measurements can be directly injected. In this way the triggering conditions can be locally checked. In order to be able to obtain such estimators we also need to relax the triggering condition (4) in the local nodes.

$$\begin{aligned} \sigma_M(Q)\|e\|^2 &\leq (\sigma_m(L - S)\|x\|^2 + u_k^T u_k) \\ \Rightarrow \dot{V} &\leq -x^T Sx \end{aligned} \quad (18)$$

Now, as in the design of the self-triggered estimator, we can obtain estimators with direct injection of local measurements for $\|e\|$ and $\|x\|$. Let us first partition and reorder the state-space as $e^i = [e_i, \hat{e}^i]^T$, with e_i denoting the local error variable at node i , and \hat{e}^i the remaining entries of the error vector. With this new description of the error vector at node i , we can rewrite the dynamics for e as

$$\begin{aligned} \frac{d}{dt} \begin{bmatrix} e_i \\ \hat{e}^i \end{bmatrix} &= \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} e_i \\ \hat{e}^i \end{bmatrix} \\ &- \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} x_i(t_k) \\ \hat{x}^i(t_k) \end{bmatrix} - \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u_k \end{aligned}$$

Then,

$$\begin{aligned} \frac{d}{dt} \|\hat{e}^i\| &\leq \|A_3 e_i + A_4 \hat{e}^i - B_2 u_k - [A_3 \ A_4] x_k^i\| \leq \\ &\|A_4\| \|\hat{e}^i\| + (\|A_3 e_i - B_2 u_k\|^2 + \|[A_3 \ A_4]\|^2 \|x_k\|^2 \\ &\quad + 2\|(A_3 e_i - B_2 u_k)^T [A_3 \ A_4]\| \|x_k\|)^{\frac{1}{2}} \end{aligned}$$

and hence we can use as bound estimator:

$$\begin{aligned} e^T e &\leq e_i^2 + (\varepsilon^i)^2 & (19) \\ \frac{d}{dt} \varepsilon^i &= \|A_4\| \varepsilon^i + (\|A_3 e_i - B_2 u_k\|^2 \\ &\quad + \|[A_3 \ A_4]\|^2 \|x_k\|^2 \\ &\quad + 2\|(A_3 e_i - B_2 u_k)^T [A_3 \ A_4]\| \|x_k\|)^{\frac{1}{2}} \end{aligned} \quad (20)$$

In an analogous way we can derive a lower bound estimator for $\|x\|^2$ resulting in:

$$x^T x \geq x_i^2 + (\chi^i)^2 \quad (21)$$

$$\frac{d}{dt} \chi^i = -\|A_4\| \chi^i - \|A_3 x_i + B_2 u_k\| \quad (22)$$

Therefore, local triggering conditions of the form

$$\sigma_M(Q) (e_i^2 + (\varepsilon^i)^2) \leq \sigma_m(L - S) (x_i^2 + (\chi^i)^2) + u_k^T u_k \quad (23)$$

could be used to decide the update times. The distributed algorithm proposed in section IV-A will enforce a new update only when all nodes have their local rules violated. Therefore, the distributed algorithm will ensure that the controller follows the node with least conservative estimates, and thus the system waits as long as possible before triggering a new update. Now we can summarize the event-triggered protocol in the following proposition:

Proposition 5.2: The distributed implementation of a LQR controller (2), designed to stabilize the linear system (1), defined by the algorithm described in Section IV with update times given by (23), (20), and (22) guarantees stability of the closed loop system and enforces $\dot{V} \leq -x^T S x$. Moreover, the algorithm is energy-balanced.

VI. EXAMPLE

We illustrate these techniques in two small examples. Let us start with the system defined by

$$A = \begin{bmatrix} 1 & -2 & 0 & 2 & 0 \\ 2 & -4 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 3 \end{bmatrix}, B = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \\ 2 \end{bmatrix}, x_o = \begin{bmatrix} 1 \\ -2 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

The controller was designed solving the Riccati equation (3), with $L = 0.5I$ and setting $K = -B^T P$. The S matrix was set to $S = 0.3I$. For comparison we include plots of simulations using the centralized event-triggered rule $e^T Q e - u_k^T u_k \leq x^T (L - S) x$, which does not involve any approximation or relaxation from (4). This protocol would be giving us the exact times when the controller needs to be updated. Obviously our two protocols for networks will give us more updates, but not necessarily always smaller times.

In figure 1 we can see how the three strategies achieve the desired performance as measured by the decay of the

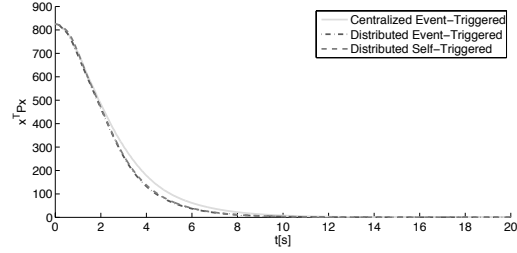


Fig. 1. Lyapunov function decay for centralized event-triggered, distributed event-triggered and decentralized self-triggered. First example.

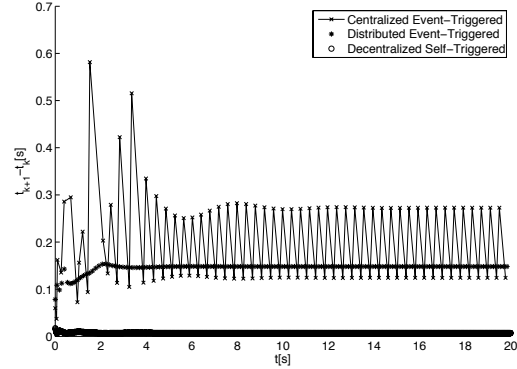


Fig. 2. Inter-sample times for centralized event-triggered, distributed event-triggered and decentralized self-triggered strategies for the first example.

Lyapunov function. But looking at figure 2, while the times obtained with the self-triggered strategy are much shorter than the centralized event-triggered controller, the distributed event-triggered strategy produces times much closer to those from the centralized controller. This effect is explained by the fact that in an event-triggered protocol measurements from the plant at a local level are taken continuously, while the self-triggered protocol is completely open loop between updates.

Finally we present a comparison of the energy consumption evolution at a node (and therefore at all of them as our algorithms are balanced). The values for the different costs used in the energy computation were: $C_1 = 38.4mJ$, $C_2 = 3.2mJ$, $l = 60mW$, $d_2 = 24mW$ and $D_1 = 0mJ$ negligible. These values are approximations obtained from the actual power consumption values for a MicaZ [23] using ZigBee [24] for wireless communication. Figure 3 depicts the energy consumption using the self-triggered and the event-triggered protocols, as given by (6) and (7). Observe how the distributed event-triggered algorithm consumes less energy than the self-triggered. In this example the event-triggered strategy becomes more energy-efficient because of the great difference on number of updates between event and self-triggered. But in general that does not have to be the case. Even if the event-triggered algorithm produces fewer updates than the self-triggered, the first algorithm could consume more energy. The next example tries to portray this situation.

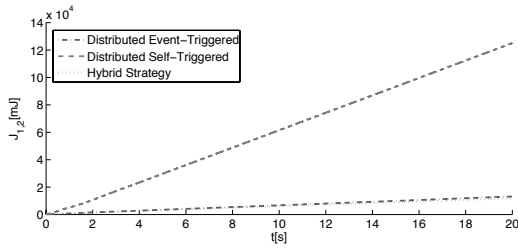


Fig. 3. Energy consumption for distributed event-triggered and decentralized self-triggered strategies for the first example.

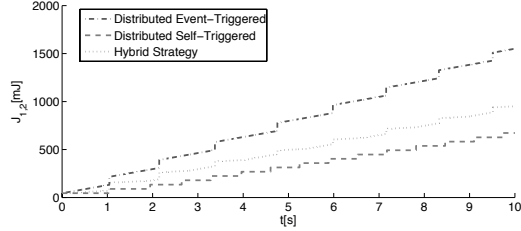


Fig. 4. Energy consumption for distributed event-triggered and decentralized self-triggered strategies for the second example.

The system is described by:

$$A = 10^{-2} \begin{bmatrix} -10 & -5 & 11 & -8 & 3 \\ -7 & -9 & 14 & -16 & 7 \\ -5 & -7 & 11 & -15 & 7 \\ -6 & -7 & 10 & -12 & 6 \\ -13 & -8 & 16 & -11 & 3 \end{bmatrix}, B = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.1 \\ 0.2 \end{bmatrix},$$

$$x_o = [-4.8 \ -11 \ -5.3 \ -2.9 \ -3.7]^T$$

and we used again $L = 0.5I$ and $S = 0.3I$.

In figure 4 we can appreciate how the self-triggered protocol again produces more frequent updates, but if we look at the energy consumption we can see how the distributed event-triggered algorithm consumes more energy all the time. In this example the evolution of the system is quite slow, and therefore the listening and computing energy plays a bigger role, making the distributed event-triggered strategy less efficient. A hybrid strategy in which the event-triggered protocol also uses the self-triggered predicted times to keep the radio-module asleep is also presented in both figure 3 and figure 4. This algorithm improves the performance of the event-triggered algorithm but does not always outperform the self-triggered algorithm.

VII. DISCUSSION

We have proposed two distributed implementations for controllers over SAN's based on the event-triggered and self-triggered control paradigms. The use of event-triggered and self-triggered control is justified in terms of the reduction in energy consumption obtained by reducing the number of messages exchanged over the network while achieving the prescribed control performance. Both strategies were illustrated through two examples which show that no algorithm outperforms the other in terms of energy expenditure. A

hybrid strategy benefiting from both approaches was tested. This hybrid protocol always outperforms the event-triggered approach, but that is not the case with the self-triggered protocol. The selection of the most adequate protocol for a given control system is left for future research. Note that the results obtained strongly depend on the particular estimators selected. We are currently investigating how to improve the performance of the proposed estimators.

REFERENCES

- [1] J. Fisher, T. Harmon, and W. Kaiser, "Multiscale river hydraulic and water quality observations combining stationary and mobile sensor network nodes," in *American Geophysical Union Joint Assembly Annual Spring Meeting*, Baltimore, MD, May 23-26 2006.
- [2] L. Girod and M. Roch, "An overview of the use of remote embedded sensors for audio acquisition and processing," in *In the Proceedings of the IEEE International Symposium on Multimedia (ISM)*, San Diego, CA, December 2006.
- [3] S. Oh and S. Sastry, "Tracking on a graph," in *Information Processing in Sensor Networks (IPSN)*, 2005.
- [4] B. Sinopoli, C. Sharp, L. Schenato, S. Shaffert, and S. Sastry, "Distributed control applications within sensor networks," in *Proceedings of the IEEE, Special Issue on Sensor Networks and Applications*, 2005.
- [5] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Information Processing in Sensor Networks (IPSN)*, 2004.
- [6] J. Djughash, S. Singh, and B. Grocholsky, "Decentralized mapping of robot-aided sensor networks," in *IEEE International Conference on Robotics and Automation*, 2008.
- [7] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Information Processing in Sensor Networks (IPSN)*, 2006.
- [8] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Multi-hop communication in order-optimal for homogeneous sensor networks," in *Information Processing in Sensor Networks (IPSN)*, 2004.
- [9] H. Le, D. Henriksson, and T. Abdelzaher, "A control theory approach to throughput optimization in multi-channel collection sensor networks," in *Information Processing in Sensor Networks (IPSN)*, 2007.
- [10] M. Paskin, C. Guestrin, and J. McFadden, "A robust architecture for distributed inference in sensor networks," in *Information Processing in Sensor Networks (IPSN)*, 2005.
- [11] R. Subramanian and F. Fekri, "Sleep scheduling and lifetime maximization in sensor networks: Fundamental limits and optimal solutions," in *Information Processing in Sensor Networks (IPSN)*, 2006.
- [12] G. Franklin, "Rational rate," in *IEEE Control Systems Magazine*, 2007.
- [13] G. Goodwin, S. Graebe, and M. Salgado, *Control System Design*. Prentice Hall, 2001.
- [14] C. Houpis and G. B. Lamont, *Digital Control Systems*. McGraw-Hill Higher Education, 1984.
- [15] K. Åström and B. Bernhardsson, "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, 2002.
- [16] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," in *IEEE Transactions on Automatic Control*, vol. 52(9), 2007.
- [17] X. Wang and M. D. Lemmon, "State based self-triggered feedback control systems with l2 stability," in *17th IFAC World Congress*, 2008.
- [18] A. Anta and P. Tabuada, "Self-triggered stabilization of homogeneous control systems," in *American Control Conference*, 2008.
- [19] X. Wang and M. D. Lemmon, "Event-triggered broadcasting across distributed networked control systems," in *American Control Conference*, 2008.
- [20] M. Velasco, J. Fuertes, and P. Marti, "The self triggered task model for real-time control systems," *Work in Progress Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS WIP 2003)*, pp. 67-70, 2003.
- [21] A. Tel, *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.
- [22] H. Khalil, *Nonlinear systems*. Prentice Hall Upper Saddle River, NJ, 2002.
- [23] [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless.pdf/6020-0060-01_A.MICAz.pdf
- [24] [Online]. Available: <http://www.zigbee.org>