2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

A PI Controller based on Asymmetric Gossip Communications for Clocks Synchronization in Wireless Sensors Networks

Ruggero Carli Edoardo D'Elia Sandro Zampieri

Abstract— **In this paper a distributed clock synchronization algorithm is proposed. The algorithm requires asymmetric gossip communications between the nodes of the network, and is based on an PI-like consensus protocol where the proportional part compensates the different clock speeds while the integral part eliminates the different clock offsets. Convergence of the algorithm is proved and analyzed with respect to the controller parameter, when the underlying graph is the complete graph. Simulations results show the effectiveness of the proposed strategy also for more general communication topologies.**

I. Introduction

The recent technological advances in wireless communication and the decreasing in cost and size of electronic devices have promoted the appearance of large inexpensive interconnected systems, each with computational and sensing capabilities. These complex networks of agents are used in a large number of applications covering a wide range of fields, such as, surveillance, targeting systems, controls, communications, monitoring areas, intrusion detection, vehicle tracking and mapping. One key problem in many of these applications is clock-synchronization. Indeed, very often, it is essential that the agents act in a coordinated and synchronized fashion requiring global clock synchronization, that is, all the agents of the network need to refer to a common notion of time. A wide variety of clock synchronization protocols have been proposed recently in the literature. Depending upon the architectures adopted, these protocols can be divided into three categories: tree-structure-based, cluster-structure-based, and fully-distributed.

Tree-structure-based protocols [1], [2] consist in electing a reference node and creating a spanning tree rooted at this reference node, where each children synchronizes itself with respect to its parent. In cluster-structure-based protocols [3], the network is divided into distinct clusters, each with an elected cluster-head. All nodes within the same cluster synchronize themselves with the corresponding cluster-head, and each cluster-head synchronizes itself with an another cluster-head. Although these two strategies have been experimentally tested showing remarkable performance, they suffer from robustness and scalability issues. For instance, if a node dies, then it is necessary to rebuild the tree or the clusters, at the price of additional implementation overhead and possibly long periods in which the network is poorly synchronized.

The research leading to these results has been supported by the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement n. FP7-ICT-223866- FeedNetBack and under grant agreement n. 257462 HYCON2 Network of Excellence.

Ruggero Carli, Edoardo D'Elia and Sandro Zampieri are with the Department of Information Engineering, University of Padova, Via Gradenigo 6/a, 35131 Padova, Italy {carlirug|deliaedo|zampi}@dei.unipd.it.

Fully distributed algorithms for clocks synchronization have appeared in [4], [5], [6], [7]. The authors in [5] introduced a protocol inspired by the fireflies integrate-and-fire synchronization mechanism, able to compensate for different clock offsets but not for different clock skews. On the opposite, the algorithm proposed in [4] adopting a P-controller, compensates for the clock skews but not for the offsets. Distributed protocols that can compensate for both clock skews and offsets have been proposed in [6], [7]. The first one is based on the cascade of two distributed least-squared algorithms, while the second one is based on the cascade of two first order consensus algorithms [8], [9]. Of note is the fact that both these strategies are highly non-linear and do not lead to a simple characterization of the effects of noise on the steady-state performance.

Differently, [10], [11] proposed a synchronization algorithm that can be formally analyzed not only in the noiseless scenario in terms of rate of convergence but also in a noisy setting in terms of the steady-state synchronization error. This algorithm compensates for both initial offsets and differences in internal clock speeds and is based on a Proportional-Integral (PI) controller that treats the different clock speeds as unknown constant disturbances and the different clock offsets as different initial conditions for the system dynamics. Both convergence guarantees as well optimal design using standard optimization tools when the underlying communication graph is known, can be provided [12]. It is important to remark that the time-synchronization algorithm proposed by [11] requires each node to perform all the operations related to the k-th iteration of the algorithm, including transmitting messages, receiving messages and updating estimates, within a short time window. This pseudo-synchronous implementation might be very sensitive to packet losses, node and link failure.

In this paper we developed and analyzed a far more practical version of the PI synchronization algorithm, assuming that clocks can communicate by an asymmetric gossip protocol. That is, at each iteration only one node can establish a directional communication with only one of its neighbors. This applies very well to real sensor networks, and drastically reduces the network requirements in terms of reliability, bandwidth, and synchronization. Theoretical results are provided when the underlying communication topology is given by the complete graph, while more general families of graphs are considered by means of simulations.

A. Mathematical preliminaries

Before proceeding, we collect some useful definitions and notations. In this paper, $\mathcal{G} = (V, \mathcal{E})$ denotes a *directed graph* where $V = \{1, \dots, N\}$ is the set of vertices and

$\mathcal{E}$ is the set of directed edges, i.e., a subset of $V \times V$. We assume that no self-loops are contained in $\mathcal{E}$. Given a node i, by $\mathcal{N}_i$ we denote the neighbors of i, namely, $\mathcal{N}_i = \{j \in V \,|\, (j,i) \in \mathcal{E}, i \neq j\}$. Moreover, by d_i we denote the cardinality of $\mathcal{N}_i$, i.e., $d_i = |\mathcal{N}_i|$. Given a vector $v \in \mathbb{R}^N$ and a matrix $M \in \mathbb{R}^{N\times N}$, we let v^* and M^* respectively denote the transpose of v and of M. With the symbols $\mathbb{1}$ we denote the N-dimensional vector having all the components equal to 1. Given $v = [v_1, \ldots, v_N]^* \in \mathbb{R}^N$, $\mathrm{diag}\{v\}$ or $\mathrm{diag}\{v_1, \ldots, v_N\}$ mean a diagonal matrix having the components of v as diagonal elements. The vector $e_i \in \mathbb{R}^N$ denotes the i-th vector of the canonical basis, i.e., $e_i = [0, \ldots, 0, 1, 0, \ldots, 0]^*$ with the i-th component equal to 1. The matrix $I \in R^{N\times N}$ denotes the N-dimensional identity while with the symbol Ω we denote the N-dimensional matrix $I - \frac{1}{N}\mathbb{1}\mathbb{1}^*$.

II. Problem Formulation

A. Mathematical modeling of a clock

We start by proposing a model of each local clock which is simple enough but which captures the main difficulty of the problem we want to solve, namely the fact that the time is an unknown variable, which has to be estimated.

Assume that each unit has a clock, which is an oscillator capable to produce an event at time $t(k)$, $k = 0, 1, 2, \ldots$ The clock has to use these clicks in order to estimate the time. The following cumulative function well describes the time evolution of the tick counter that can be implemented in the clock

$$s(t) = \int_{-\infty}^{t} f(\sigma) d\sigma,$$

where

$$f(t) = \sum_{k=0}^{\infty} \delta(t - t(k)). \tag{II.1}$$

In this way the counter output is a step shaped function. In case the clock period is small, in order to simplify the analysis, it is convenient to approximate step shaped function $s(t)$ with a continuous one by approximating the delta shaped function $f(t)$ with a regularized version. One way is to take

$$f(t) := \frac{1}{t(k+1) - t(k)} \quad \text{for all} \quad t \in [t(k), t(k+1)[\tag{II.2}$$

The graphs of the stepwise and of the regularized version of $s(t)$ is shown in Figure 1. Notice that $f(t)$ can be interpreted as the oscillator frequency at time t. Typically an estimate $\hat{f}$ of $f(t)$ is available and it is known that $f(t) \in [f_{\min}, f_{\max}]$. From the counter one can build a time estimate $\hat{t}(t)$ by letting

$$\hat{t}(t) = \hat{t}(t_0) + \hat{\Delta}(t)[s(t) - s(t_0)] = \hat{t}(t_0) + \hat{\Delta}(t) \int_{t_0}^{t} f(\sigma)\, d\sigma \tag{II.3}$$

where $\hat{\Delta}(t)$ is an estimate of the oscillation period $1/f(t)$. It is reasonable to initialize $\hat{\Delta}(t)$ to $1/\hat{f}$.

Both $\hat{t}(t)$ and $\hat{\Delta}(t)$ can be modified when the unit obtains information allowing it to improve its time and oscillator frequency estimates. Assume that these corrections are applied at time instants $T_{\mathrm{up}}(h)$, where $h = 0, 1, \ldots$, called updating time instants. In this case we have [1]

$$\begin{cases} \hat{t}(T^+_{\mathrm{up}}(h)) = \hat{t}(T^-_{\mathrm{up}}(h)) + u'(h) \\ \hat{\Delta}(T^+_{\mathrm{up}}(h)) = \hat{\Delta}(T^-_{\mathrm{up}}(h)) + u''(h) \end{cases}$$

where u' and u'' denote the control inputs applied to $\hat{t}$ and $\hat{\Delta}$, respectively. For $t \in \left[T^+_{\mathrm{up}}(h), T^-_{\mathrm{up}}(h+1)\right]$ the estimates evolve according to

$$\begin{cases} \hat{t}(t) = \hat{t}(T^+_{\mathrm{up}}(h)) + \hat{\Delta}(T^+_{\mathrm{up}}(h))\,(s(t) - s(T_{\mathrm{up}}(h))) \\ \hat{\Delta}(t) = \hat{\Delta}(T^+_{\mathrm{up}}(h)) \end{cases} \tag{II.4}$$

or according to

$$\begin{cases} \hat{t}(t) = \hat{t}(T^+_{\mathrm{up}}(h)) + \hat{\Delta}(T^-_{\mathrm{up}}(h))\,(s(t) - s(T_{\mathrm{up}}(h))) \\ \hat{\Delta}(t) = \hat{\Delta}(T^+_{\mathrm{up}}(h)) \end{cases} \tag{II.5}$$

Intuitively the updating rule (II.4) should perform better than the updating rule (II.5). However, in this paper we assume that the units adopt the updating rule (II.5). We will clarify later in Remark 5.2 the reasons of this choice.

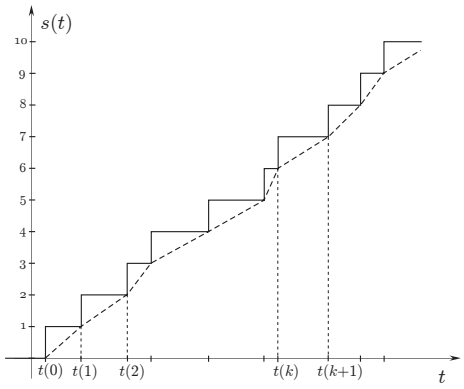

Fig. 1. The graphs of of $s(t)$ resulting from the impulse shaped version of $f(t)$ given in (II.1) (continuous line) and from the regularized version of $f(t)$ given in (II.2) (dashed line).

B. Clock synchronization

Assume now that we have a network composed by N clocks. For $i \in \{1, \ldots, N\}$, let $f_i(t)$ be the evolution of the oscillator frequency of the clock i. Moreover, for $i \in \{1, \ldots, N\}$, let $x_i(t) = [x'_i(t)\ x''_i(t)]^* = [\hat{t}_i(t)\ \hat{\Delta}_i(t)]^*$ denote the local state of the clock i. The objective is to synchronize the variables $x'_i(t)$, $i \in \{1, \ldots, N\}$, namely, to find a law which allows the clocks to obtain the same time estimate.

Assume that the clocks can exchange their local state according to a graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \ldots, N\}$ and where $(i,j) \in \mathcal{E}$ whenever the clock i can send its state x_i to the clock j. Specifically, each clock i, $i \in \{1, \ldots, N\}$, transmits its state $x_i(t)$ at some time instants $T_{\mathrm{tx},i}(h)$, $h = 0, 1, \ldots$, and can use any information it receives from the neighboring nodes to perform a control at the time instants $T_{\mathrm{up},i}(h)$, $h = 0, 1, \ldots$. More precisely

$$x_i(T^+_{\mathrm{up},i}(h)) = x_i(T^-_{\mathrm{up},i}(h)) + u_i(h), \tag{II.6}$$

[1] Given the time t, with the symbols t^+ and t^- we mean, respectively, the time instant just after t and time instant just before t.

where $u_i(h) = [u'_i(h)\ u''_i(h)]^*$ is the control action applied at time $T_{\mathrm{up},i}(h)$. Moreover for $t \in \left[T^+_{\mathrm{up},i}(h), T^-_{\mathrm{up},i}(h+1)\right]$ we assume that the state x_i is updated according to (II.5).

For simplicity in this paper we assume the following properties.

Assumption 2.1: The oscillator frequencies f_i are constant, i.e., for $i \in \{1, \dots, N\}$, $f_i(t) = \bar{f}_i$ for all $t \in \mathbb{R}_{>0}$.

Assumption 2.2: The transmission delays are negligible, namely, if clock i perform its h-th transmission to clock j, then clock j receives the information $x_i(T_{\mathrm{tx},i}(h))$ exactly at time $T_{\mathrm{tx},i}(h)$ [2].

From Assumption 2.1 and from (II.3), it follows that (II.5), for the i-th clock, can be equivalently rewritten as

$$\begin{cases} x'_i(t) = x'_i(T^+_{\mathrm{up}}(h)) + x''_i(T^-_{\mathrm{up}}(h))\, \bar{f}_i\, (t - T_{\mathrm{up}}(h)) \\ x''_i(t) = x''_i(T^+_{\mathrm{up}}(h)) \end{cases} \tag{II.7}$$

The objective is to find a control strategy yielding the clock synchronization, namely such that there exist constants $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}$ such that synchronization errors

$$e_i(t) := x'_i(t) - (at + b), \quad i = \{1, \dots, N\} \tag{II.8}$$

converge to zero or remain small.

III. PI Controller based on Asymmetric Gossip Communications

To properly describe the control law we propose in this paper, we first need to introduce the data transmission and communication models the clocks adopt to exchange information with each other. Informally, we assume that the transmission's time instants are the sample times of N independent Poisson processes having all the same intensity and that the nodes communicate with each other through an *asymmetric gossip communication protocol*. In more formal terms, the data transmission and communication models are described as follows

- for $i \in \{1, \dots, N\}$, the time instants $T_{\mathrm{tx},i}(h)$, $h = 0, 1, \dots$ are the samples time of a Poisson process of intensity $\lambda > 0$;
- for $i \in \{1, \dots, N\}$ and for $h \in \mathbb{N}$, node i sends, at time $T_{\mathrm{tx},i}(h)$, only the information related to the first component of its state, i.e., $x'_i(T_{\mathrm{tx},i}(h))$;
- for $i \in \{1, \dots, N\}$ and for $h \in \mathbb{N}$, the information $x'_i(T_{\mathrm{tx},i}(h))$ is sent by node i to only one of its neighbors, which is randomly selected with probability $1/d_i$ within the set $\mathcal{N}_i$.

Now, without loss of generality, assume that node i transmits, at time $T_{\mathrm{tx},i}(h)$, the information $x'_i(T_{\mathrm{tx},i}(h))$ to node j. Based on the information received, node j instantaneously applies to its current state $x_j(T_{\mathrm{tx},i}(h))$ the following correction

$$u = \begin{bmatrix} u' \\ u'' \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ \alpha \end{bmatrix} \left(x'_i(T_{\mathrm{tx},i}(h)) - x'_j(T_{\mathrm{tx},i}(h)) \right)$$

[2]In general, the information $x_i(T_{\mathrm{tx},i}(h))$ is received by clock $j \in N_i$ at a delayed time $T_{\mathrm{rx},i,j}(h) \geq T_{\mathrm{tx},i}(h)$ where $T_{\mathrm{rx},i,j}(h) = T_{\mathrm{tx},i}(h) + \gamma_{i,j}(h)$ being $\gamma_{i,j}(h)$ a nonnegative real number representing the deliver delay between i and j.

where α is a parameter control such that $\alpha > 0$. From (II.6), it follows that

$$\begin{aligned} x'_j(T^+_{\mathrm{tx},i}(h)) &= \frac{1}{2} \left(x'_j(T_{\mathrm{tx},i}(h)) + x'_i(T_{\mathrm{tx},i}(h)) \right) \\ x''_j(T^+_{\mathrm{tx},i}(h)) &= x''_j(T_{\mathrm{tx},i}(h)) + \\ &\quad + \frac{\alpha}{2} \left(x'_i(T_{\mathrm{tx},i}(h)) - x'_j(T_{\mathrm{tx},i}(h)) \right) \end{aligned} \tag{III.1}$$

Observe that, according to the above model, we have that $T_{\mathrm{tx},i}(h)$ represents an update time instant for node j, i.e., $T_{\mathrm{tx},i}(h) = T_{\mathrm{up},j}(h')$ for some $h' \in \mathbb{N}$.

Remark 3.1: Observe that the control above introduced can be seen as a PI controller where $u' = x'_i(T_{\mathrm{tx},i}(h)) - x'_j(T_{\mathrm{tx},i}(h))$ and $u'' = \alpha \left(x'_i(T_{\mathrm{tx},i}(h)) - x'_j(T_{\mathrm{tx},i}(h)) \right)$ represent, respectively, the proportional and the integral part. It is worth mentioning that this strategy has been inspired by the PI consensus controller strategy proposed in [10]. However, in [10] the synchronization protocol is analyzed in its synchronous implementation, i.e., assuming that all the nodes perform the transmitting and updating actions synchronously. The aim of this paper is to adapt the PI synchronization algorithm to the more practical scenario where the nodes are assumed to exchange information through an asymmetric gossip communication protocol.

Now let $\{T_{\mathrm{up}}(h), h \in \mathbb{N}\}$ be the set of all the updating time instants of the clocks' network, i.e.,

$$\{T_{\mathrm{up}}(h), h \in \mathbb{N}\} = \bigcup_{i=1}^{N} \{T_{\mathrm{up},i}(h), h \in \mathbb{N}\}.$$

Notice that, for any $h \in \mathbb{N}$, there exist $i, j \in \{1, \dots, N\}$ and $h', h'' \in \mathbb{N}$ with $h' \leq h$, $h'' \leq h$, such that

$$T_{\mathrm{up}}(h) := T_{\mathrm{up},j}(h') = T_{\mathrm{tx},i}(h'').$$

Moreover observe that, since the N Poisson processes generating the transmission's time instants are independent from one another, the updating time instants $\{T_{\mathrm{up}}(h), h \in \mathbb{N}\}$ can be seen as the sample times of a Poisson process of intensity $N\lambda$.

Next we provide a convenient vector-form description of the evolution of the clocks' network. To do so, we need some auxiliary definitions. First, let us introduce

$$x' := [x'_1, \dots, x'_N]^* \in \mathbb{R}^N, \quad x'' := [x''_1, \dots, x''_N]^* \in \mathbb{R}^N$$
$$x = [x'_1, \dots, x'_N, x''_1, \dots, x''_N]^* \in \mathbb{R}^{2N}$$

Second, for $i, j \in \{1, \dots, N\}$ let the matrix $E_{i \to j} \in \mathbb{R}^{N \times N}$ be defined as

$$E_{i \to j} := e_j e_j^* - e_j e_i^*$$

and let the matrix $D \in \mathbb{R}^{N \times N}$ be defined as

$$D = \mathrm{diag}\{\bar{f}_1, \dots, \bar{f}_N\}.$$

Finally let $\delta T_{\mathrm{up}}(h) := T_{\mathrm{up}}(h+1) - T_{\mathrm{up}}(h)$.

Now, without loss of generality, assume that $T_{\mathrm{up}}(h) := T_{\mathrm{up},j}(h') = T_{\mathrm{tx},i}(h'')$ for some $h', h'' \in \mathbb{N}$. Then, combining (II.7) with (III.1), we can write

$$x(T^-_{\mathrm{up}}(h+1)) = \begin{bmatrix} I - \frac{1}{2} E_{i \to j} & \delta T_{\mathrm{up}}(h) D \\ -\frac{\alpha}{2} E_{i \to j} & I \end{bmatrix} x(T^-_{\mathrm{up}}(h))$$

To simplify the notation we define

$$x(h) := x(T_{\rm up}^-(h)).$$

Hence the above system becomes

$$x(h+1) = \begin{bmatrix} I - \frac{1}{2}E_{i\to j} & \delta T_{\rm up}(h) D \\ -\frac{\alpha}{2}E_{i\to j} & I \end{bmatrix} x(h). \qquad \text{(III.2)}$$

We extensively simulated algorithm in (III.2), for several communication graphs $\mathcal{G}$ and for different values of D, λ and α. Next we summarize some of the numerical evidences we extrapolated from the simulations we run:

a) Typically, for $\alpha > \lambda/2$, the algorithm does not reach the synchronization, independently from the values of $\bar{f}_i$.
b) For $\alpha \leq \lambda/2$ the synchronization is achieved depending on how spread the values $\{\bar{f}_i\}_{i=1}^N$ are. More precisely suppose that, for $i \in \{1,\dots,N\}$, $\bar{f}_i \in [1-\epsilon\ 1+\epsilon]$ where $0 \leq \epsilon < 1$. Then the smaller the value of α is, the greater the value of ϵ is while still reaching the synchronization. In particular, if $\alpha \approx \lambda/2$ then the synchronization is attained only if $\epsilon << 1$, while if $\alpha \approx 0$ then ϵ can be also very close to 1.
c) When the synchronization is achieved, then $\lim_{h\to\infty} x''(h) = \beta \mathbb{1}$ where β is in general close to $\frac{1}{N}\sum_{i=1}^N \bar{f}_i$. This value β represents the oscillator frequency of the "virtual clock" to which all the clocks synchronize.

Providing a theoretical analysis of the above evidences is quite challenging in general. In the next section we restrict to $\mathcal{G}$ being the complete graph. In this case we will be able to provide some theoretical insights on the convergence properties of algorithm (III.2).

IV. Mean-Square Analysis

To our aims it is convenient to consider the synchronization error $y(h) = \Omega x'(h)$ and the new variable $z(h) = \Omega D x''(h)$. Since for any $i,j \in \{1,\dots,N\}$, $i \neq j$, $E_{i\to j}\Omega = E_{i\to j}$, system (III.2) can be rewritten as

$$\begin{bmatrix} y(h+1) \\ z(h+1) \end{bmatrix} = \begin{bmatrix} I - \frac{1}{2}\Omega E(h) & \delta T_{\rm up}(h) \\ -\frac{\alpha}{2}\Omega D E(h) & I \end{bmatrix} \begin{bmatrix} y(h) \\ z(h) \end{bmatrix} \qquad \text{(IV.1)}$$

where $E(h) = E_{i\to j}$ if, during the h-th iteration node i and node j are, respectively, the transmitting and the receiving nodes. Clearly x' reaches the asymptotic synchronization if and only if $\lim_{h\to\infty} y(h) = 0$. Now observe that, according to the data transmission and communication model described in Section (III), system (IV.1) evolves as a random process such that

- for any $h \in \mathbb{N}$, the matrix $E(h)$ is randomly selected within the set

$$\mathcal{S} = \{E_{i\to j} \,|\, (i,j) \in \mathcal{E}\}$$

and $\mathbb{P}\left[E(h) = E_{i\to j}\right] = \frac{1}{N\, d_i}$;
- the selection of the matrix $E(h)$ is independent from the selection of $E(h')$, $h' \neq h$;
- $\{\delta T_{\rm up}(h) |\ h \in \mathbb{N}\}$ are the interarrival times of a Poisson process of intensity $N\lambda$.

The goal is to perform a mean-square analysis of (IV.1). To do so, we introduce the matrix

$$\Sigma(h) := \mathbb{E}\left[\begin{bmatrix} y(h) \\ z(h) \end{bmatrix} \begin{bmatrix} y^*(h) & z^*(h) \end{bmatrix}\right] = \begin{bmatrix} \Sigma_{yy}(h) & \Sigma_{yz}(h) \\ \Sigma_{yz}^*(h) & \Sigma_{zz}(h) \end{bmatrix},$$

where

$$\begin{aligned} \Sigma_{yy}(h) &:= \mathbb{E}[y(h)y^*(h)], \\ \Sigma_{yz}(h) &:= \mathbb{E}[y(h)z^*(h)], \\ \Sigma_{zz}(h) &:= \mathbb{E}[z(h)z^*(h)]. \end{aligned}$$

The objective is to study the evolution of

$$\Sigma(h+1) = \mathbb{E}[A(h)\Sigma(h)A^*(h)] \qquad \text{(IV.2)}$$

where

$$A(h) := \begin{bmatrix} I - \frac{1}{2}\Omega E(h) & \delta T_{\rm up}(h) \\ -\frac{\alpha}{2}\Omega D E(h) & I \end{bmatrix}.$$

In what follows, we perform our analysis by assuming that D is a small perturbation of the matrix I. Accordingly, we will design the parameter α only for $D = I$. From the fact that the eigenvalues of the expectation operator in (IV.2) depend continuously on the matrix D, it will follow that this choice of α yields the stability also for a small enough perturbation of D. However, we will come back on the robustness of our algorithm with the respect to different values of oscillator frequencies in the numerical Section VI.

Assuming that $D = I$, from (IV.2) we obtain the following recursive equations[3]

$$\begin{aligned} \Sigma_{yy}^+ &= \Sigma_{yy} - \frac{1}{2}\left\{\Omega\mathbb{E}[E_{i\to j}]\Sigma_{yy} + \Sigma_{yy}\mathbb{E}[E_{i\to j}^*]\Omega\right\} \\ &\quad + \frac{1}{4}\Omega\mathbb{E}[E_{i\to j}\Sigma_{yy}E_{i\to j}^*]\Omega + \frac{1}{N\lambda}\left\{\Sigma_{yz} + \Sigma_{yz}^*\right\} \\ &\quad - \frac{1}{2N\lambda}\left\{\Omega\mathbb{E}[E_{i\to j}]\Sigma_{yz} + \Sigma_{yz}^*\mathbb{E}[E_{i\to j}^*]\Omega\right\} + \frac{2}{N^2\lambda^2}\Sigma_{zz} \\ \Sigma_{yz}^+ &= -\frac{\alpha}{2}\Sigma_{yy}\mathbb{E}[E_{i\to j}^*]\Omega + \frac{\alpha}{4}\Omega\mathbb{E}[E_{i\to j}\Sigma_{yy}E_{i\to j}^*]\Omega + \Sigma_{yz} \\ &\quad - \frac{1}{2}\Omega\mathbb{E}[E_{i\to j}]\Sigma_{yz} - \frac{\alpha}{2N\lambda}\Sigma_{yz}^*\mathbb{E}[E_{i\to j}^*]\Omega + \frac{1}{N\lambda}\Sigma_{zz}, \\ \Sigma_{zz}^+ &= \frac{\alpha^2}{4}\Omega\mathbb{E}[E_{i\to j}\Sigma_{yy}E_{i\to j}^*]\Omega \\ &\quad - \frac{\alpha}{2}\left\{\Omega\mathbb{E}[E_{i\to j}]\Sigma_{yz} + \Sigma_{yz}^*\mathbb{E}[E_{i\to j}^*]\Omega\right\} + \Sigma_{zz}. \end{aligned}$$

where we used the fact that $\mathbb{E}[\delta T_{\rm up}(h)] = \frac{1}{N\lambda}$ and $\mathbb{E}[(\delta T_{\rm up}(h))^2] = \frac{2}{N^2\lambda^2}$. The covariance matrix Σ then updates according to a linear transformation

$$\Sigma(h+1) = F[\Sigma(h)] \qquad \text{(IV.3)}$$

defined by the recursive equations that we just computed, and whose initial conditions can be obtained once we state the following assumption on $x'(0)$, $x''(0)$.

Assumption 4.1: The initial condition $x'(0)$ is a random vector such that $\mathbb{E}[x'(0)] = 0$, $\mathbb{E}[x'(0)x'(0)^*] = \sigma_x^2 I$. The vector $x''(0)$ is such that $x''(0) = \mathbb{1}$.

[3]For the sake of notational simplicity, time dependence has been omitted here; the notation Σ_{ij}^+ stand for $\Sigma_{ij}(h+1)$.

It then follows that

$$\Sigma(0) = \begin{bmatrix} \sigma_x^2\Omega & 0 \\ 0 & 0 \end{bmatrix} \tag{IV.4}$$

The analysis of the previous recursive equations is a challenging problem when $\mathcal{G}$ is an arbitrary communication graph. In the next section we provide a detailed theoretical analysis for the complete graph. In Section VI we consider also a more realistic family of graphs through numerical simulations.

A. Complete Graph

Assume that the graph $\mathcal{G}$ describing the feasible communication between nodes is the complete graph. Observe that, in this case

$$\mathbb{P}[E(h) = E_{i\to j}] = \frac{1}{N(N-1)}$$

for any $(i,j) \in \mathcal{E}$. We have the following technical Lemma.

Lemma 4.2: In the case of complete graph it holds

$$\mathbb{E}[E_{i\to j}] = \frac{1}{N-1}\Omega, \quad \text{and} \quad \mathbb{E}[E_{i\to j}\Omega E_{i\to j}^*] = \frac{2}{N}I.$$

Due to limitations of space we refer the reader to [13] for the proof of the above result and of all the results in the next section. We can now state the following results.

Proposition 4.3: In the case of complete graph, the set

$$\mathcal{J} = \left\{ \Sigma | \Sigma = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \otimes \Omega \right\}$$

is invariant under the transformation in (IV.3).

Proposition 4.4: In the case of a complete graph and under Assumption 4.1, we have

$$\begin{aligned} \Sigma_{yy}(h) &= \xi_{yy}(h)\Omega, \\ \Sigma_{yz}(h) &= \xi_{yz}(h)\Omega, \\ \Sigma_{zz}(h) &= \xi_{zz}(h)\Omega, \end{aligned}$$

where

$$\begin{bmatrix} \xi_{yy}^+ \\ \xi_{yz}^+ \\ \xi_{zz}^+ \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{2N^2-3N-1}{2N(N-1)} & \frac{2N-3}{N\lambda(N-1)} & \frac{2}{N^2\lambda^2} \\ -\frac{\alpha}{2N(N-1)} & \frac{2N^2\lambda-3N\lambda-\alpha}{2N\lambda(N-1)} & \frac{1}{N\lambda} \\ \frac{\alpha^2}{2N} & -\frac{\alpha}{N-1} & 1 \end{bmatrix}}_{\Phi} \begin{bmatrix} \xi_{yy} \\ \xi_{yz} \\ \xi_{zz} \end{bmatrix}$$

V. Algorithm Convergence

The following theorem characterizes the convergence properties of the proposed algorithm when the underlying graph is the complete graph.

Theorem 5.1: Consider the network of clocks described in Section III with an underlying complete graph. Then the variance Σ of the synchronization error converges exponentially to zero if and only if

$$\alpha < \bar{\alpha}(N)$$

where

$$\bar{\alpha}(N) := \frac{N\lambda}{N-1}\Big\{ \sqrt{N^4 - 4N^3 + 9N^2 - 8N + 3} - N^2 + 2N - 2 \Big\}$$

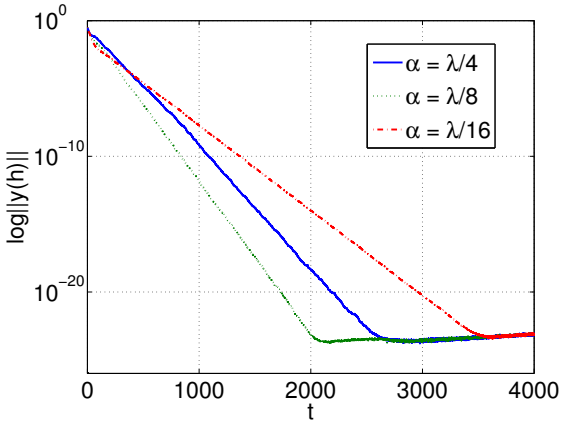

Fig. 2. Behavior of the algorithm for the complete graph topology as α changes: $\lambda/4$ (solid line), $\lambda/8$ (dotted line) and $\lambda/16$ (dash-dot line).

Remark 5.2: Instead of (II.5), we might consider the updating rule in (II.4). The algorithm resulting in this case satisfies the same properties of (IV.1). In particular, for the complete graph, it is possible to see that there still exists a function $\bar{\alpha}(N)$, such that $\bar{\alpha}(N) > \lambda/2$ for all N, $\lim_{N\to\infty} \bar{\alpha}(N) = \lambda/2$ and such that the synchronization is attained if and only if $\alpha < \bar{\alpha}$. However the analysis in this case is much more involved; for this reason, in this paper, we decide to analyze the algorithm adopting (II.5).

VI. Simulation Results

In this section we provide some numerical results illustrating the synchronization algorithm we propose in this paper. We run our simulations over two different topologies of network, namely, the complete graph and the random geometric graph. In all the simulations we considered a network of $N = 50$ clocks, and we assumed that the transmissions' time instants were generated by N independent Poisson processes of the same intensity $\lambda = 0.1$. All the random geometric graphs were connected graphs and were generated by choosing the $N = 50$ points uniformly distributed in the unit square and by connecting with an edge each pair of points at distance less than 0.15.

We report our results in Figures 2, 3, 4 and 5. In all the plots we depict the behavior of $\log \|y(h)\|$ obtained by averaging over 100 Monte Carlo runs, randomized with respect to the initial conditions and, as far as the geometric topology is concerned, also with the respect to the graph. In particular, for $i \in \{1,\dots,N\}$, $x'(0)$ has been randomly chosen within the interval $[-1\ 1]$ while $\bar{f}_i$ has been randomly chosen within the interval $[1-\epsilon\ 1+\epsilon]$, where $0 \le \epsilon \le 1$ (the values of ϵ used will be specified in the captions of the figures). In Figures 2 and 3, we analyzed the behavior of $\log \|y(h)\|$ for different values of α while keeping fixed the value of ϵ to the value 10^{-4}. One can see that the value of α heavily influences the speed of convergence to the synchronization. In particular, small values of α drastically slower down the algorithm.

In Figures 4 and 5 we analyzed the behavior of $\log \|y(h)\|$ for different values of ϵ, while keeping fixed the value of α to the value of $\lambda/8$ and $\lambda/100$ for the complete and random

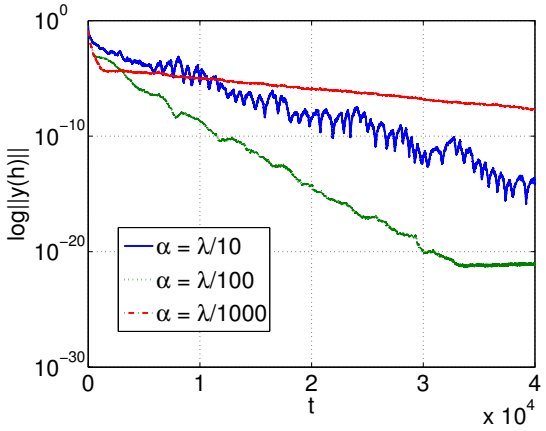

Fig. 3. Behavior of the algorithm for the random geometric graph topology as α changes: $\lambda/10$ (solid line), $\lambda/10$ (dotted line) and $\lambda/1000$ (dash-dot line).

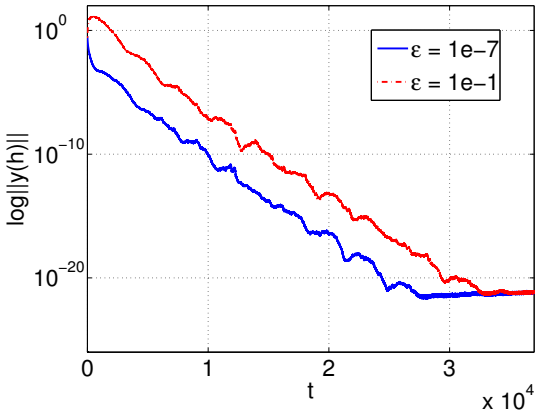

Fig. 5. Behavior of the algorithm for the random geometric graph topology as ϵ changes: 10^{-7} (solid line) and 0.1 (dash-dot line).

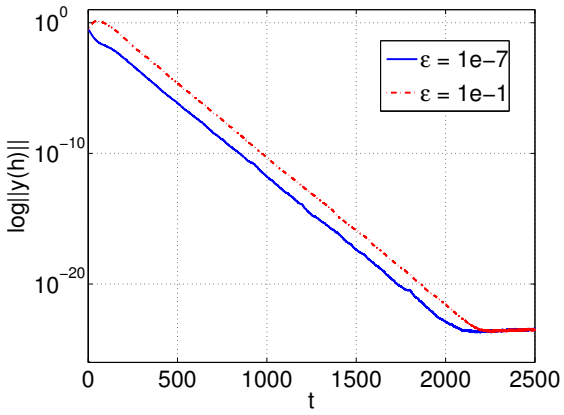

Fig. 4. Behavior of the algorithm for the complete graph topology as ϵ changes: 10^{-7} (solid line) and 0.1 (dash-dot line).

geometric graph respectively. It is remarkable that, for both the complete graph and the random geometric graph, the algorithm asymptotically achieves the synchronization, even though the drifts $\bar{f}_i$, $i \in \{1, \ldots, N\}$, are significantly spread, i.e., $\epsilon = 0.1$. However, as expected, it turns out that the smaller the value of ϵ is, the better the performance of the algorithm are.

VII. Conclusions

We developed a version of the PI algorithm that relies on an asymmetric gossip communication scheme to achieve synchronization of a network of clocks. We provided a theoretical stability analysis of the protocol, with respect to the control parameter α, if the underlying graph is the complete graph. In particular we proved that if the control parameter is under the value of $\lambda/2$, where λ represent the Poisson processes' rate, then the algorithm scales with the number of nodes. This makes the strategy independent of the network size and easier to implement in a completely distributed fashion.

Future direction of investigation include different communication graphs and protocols (e.g. multicast communication), robustness stability and performance analysis, and modeling some of the most common non-idealities like packet drops, time delivery delays and time-varying speed of the oscillators.

References

[1] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timingsync protocol for sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems (SenSys'03)*, 2003.

[2] M. Maròti, B. Kusy, G. Simon, and Àkos Ldeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys'04)*, 2004, pp. 39–49.

[3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI'02)*, 2002, pp. 147–163.

[4] O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP Journal on wireless sensor networks with coupled discrete-time oscillators*, 2007.

[5] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *ACM Conference on Embedded Networked Sensor Systems (SenSys'05)*, San Diego, November 2005.

[6] R. Solis, V. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *45th IEEE Conference on Decision and Control (CDC'06)*, San Diego, December 2006, pp. 2734–2739.

[7] L. Schenato and F. Fiorentin, "Average timesync: A consensus-based protocol for time synchronization in wireless sensor networks," in *Proceedings of 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys09)*, September 2009.

[8] R. O. Saber, J. Fax, and R. Murray, "Consensus and cooperation in multi-agent networked systems," *Proceedings of IEEE*, vol. 95, no. 1, pp. 215–233, January 2007.

[9] R. Carli, F.Fagnani, A.Speranzon, and S. Zampieri, "Communication constraints in the average consensus problem," *Automatica*, vol. 44, no. 3, pp. 671–684, 2008.

[10] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "A PI consensus controller for networked clocks synchronization," in *IFAC*, Seul, Corea, Jul. 2008.

[11] R. Carli and S. Zampieri, "Networked clock synchronization based on second order linear consensus algorithms," in *49th IEEE Conference on Decision and Control*, Atlanta, December 2010, pp. 7259–7264.

[12] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Optimal synchronization for networks of noisy double integrators," *IEEE Transactions on Automatic Control*, vol. 56, no. 5, pp. 1146–1152, 2011.

[13] R. Carli, E. D'Elia, and S. Zampieri, "A PI controller based on asymmetric gossip communications for clocks synchronization in wireless sensors networks," available at http://motion.mee.ucsb.edu/∼carlirug/Papers/ASynchro.pdf.