Network clock synchronization based on the second order linear consensus algorithm

Ruggero Carli Sandro Zampieri

Abstract

In this paper a distributed algorithm for clock synchronization is proposed. This algorithm is based on an extension of the linear consensus algorithm which is able to synchronize a family of identical double integrators. Since in reality the various clocks may have different drifts, the algorithm needs to be designed in such a way that it works correctly also in case of heterogeneous double integrators. We start by reviewing an unrealistic synchronous implementation of the clock synchronization algorithm, that has been recently proposed in the context of noisy double integrators. The main contribution of this paper is to propose a realistic *pseudo-synchronous* implementation of this algorithm. This pseudo-synchronous algorithm is shown to be a perturbation of the synchronous one and so, through arguments related the center manifold theorem, it is proved to be locally convergent under the assumption of absence of process noise, measurement noise and propagation delays. However, through numerical simulations, the performance of this algorithm is evaluated also in the case the communication delays are not negligible, the clock drifts are time-varying and the communication channels are unreliable. Moreover, again through numerical simulations, the strategy we propose in this paper is compared with other fully distributed strategies recently proposed in the literature. While being slightly slower to reach the asymptotic synchronization, our strategy seems to significantly outperform the other strategies in terms of robustness to process and measurement noises and time-varying clock drifts.

I. Introduction

One of the key problems in wireless sensor networks (WSNs) is time synchronization. Sensor networks are used in a large number of applications which cover a wide range of fields, such as, surveillance,

The research leading to these results has received funding from the European Community's Seventh Framework Program under grant agreement n. 257462 HYCON2 Network of Excellence.

R. Carli and S. Zampieri are with the Department of Information Engineering, Università di Padova, Via Gradenigo 6/a, 35131 Padova, Italy e-mail:{carlirug,zampi}@dei.unipd.it.

targeting systems, monitoring areas, intrusion detection, vehicle tracking and mapping. In many of these applications it is essential that the nodes act in a coordinated and synchronized fashion requiring global clock synchronization, that is, all the nodes of the network need to refer to a common notion of time. For instance, consider the problem of tracking a moving target using proximity sensors, where some nodes are deployed in the environment and their proximity sensors detect when the moving object passes in their vicinity [1]. Assuming the position of the sensors is known, it is essential that the instants of detection are precisely timestamped for determining the trajectory of the moving object. Clearly, the precision of the tracking algorithm based on this system is limited by the accuracy of the clock synchronization. Other interesting applications, which need a time-synchronization service, are habitat monitoring [2], power scheduling and TDMA communication schemes [3], and rapid synchronized coordination of powerlines nodes in electric power distribution networks for catastrophic power-outage prevention [4].

In designing a time synchronization algorithm for WSNs, several problems need to be considered. Indeed, WSNs are often composed by a large number of units and so the complexity of a time synchronization algorithm needs to scale well with the number of clocks in the network. Often in WSNs nodes can appear and disappear and the communication topology can change. Moreover, the communication is often unreliable and information packets can be lost due to interference. Finally, there may be time delays between the clock readings of two distinct devices. These delays are due for example to the software access time required to read the local clock, to the propagation delay, and to the reading time at the receiver clock. Concerning time delays, of note is the paper [5], where the authors characterized the fundamental limitations on synchronizing clocks over networks. In particular it is shown that determining all the unknown parameters, i.e., skews and offsets of all clocks as well as all the communication links, is impossible. Although there are algorithms that try to mitigate the effects of the delays (see e.g., [6]), the general trend is to solve this problem at hardware level by trying to read the clocks right before packet transmission and as soon the packet arrives, as discussed in [7].

A wide variety of protocols have been proposed to address the problem of time synchronization in WSNs, as surveyed in [8], [9], [10], [11]. Depending upon the architectures adopted, these protocols can be divided into three categories: tree-structure-based, cluster-structure-based, and fully distributed.

In tree-structure-based protocols a hierarchical structure is created within the network. Initially, one node is elected to be the global clock reference, then a spanning tree rooted at this node is built. Afterwards, each node synchronizes itself with its parent by compensating its offset and its relative clock drift using its parent clock readings as the direct reference. The most representative protocols in this class are the Time-synchronization Protocol for Sensor Network (TPSN) [12], the Flooding Time

Synchronization Protocol (FTSP) [7], the Lightweight Tree-based Synchronization (LTS) [13] and Tinysync [14]. Although these strategies can be easily implemented, they suffer from two main limitations. The first limitation is that they require high overhead to maintain the tree-structure and they are sensitive to network topology changes. Indeed, if the root or a non-leaf node dies or a new node is added, then it is necessary to rebuild the tree, at the price of additional implementation overhead and possibly long periods in which the network or part of it is poorly synchronized. The second limitation is that, depending on how the tree is built, it might happen that two clocks, which are physically close and can communicate with each other, belong to two different branches of the tree, thus possibly having large clock differences. This is particularly harmful in those applications such as TDMA communication which requires good synchronization of each node with its neighbors.

On the other hand, in cluster-based protocols, sensors are grouped into clusters according to their geographical locations. Within each cluster a node is elected as cluster-head. All nodes of the same cluster synchronize themselves with the corresponding cluster-head, and each cluster-head synchronizes itself with an another cluster-head. Typical protocols in this class include Pairwise Broadcast Synchronization (PBS) [15], the Reference Broadcast Synchronization (RBS) [16], Hierarchy Referencing Time Synchronization (HRTS) [17]. Like the tree-structure-based protocols, also the cluster-based protocols suffer from large overhead necessary to divide the network in cluster and to elect the reference nodes and they are somewhat sensitive to node failures.

The last approach is to have a fully distributed architecture, where there are no reference nodes and all nodes run exactly the same algorithm. These protocols have the advantage of being highly scalable and very robust to node failure and new node appearance. Indeed, recently, several fully distributed algorithms for clocks synchronization based on consensus algorithms have appeared and it is generally believed that the distributed consensus-based architectures are robust with the respect to *link and node failure*. This fact can be proved in the case the standard first order consensus algorithm is employed, for instance see [18], [19]. Concerning higher order consensus strategies, to the best of our knowledge, there are no theoretical characterizations of the above robustness properties, even though they appear to be evident in simulations. The authors in [20] 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. Alternatively, the algorithm proposed in [21], adopting a first order consensus algorithm [22], compensates for the clock skews but not for the offsets, i.e., at the end of the synchronization process all clocks will measure a constant time difference from the other clocks.

Distributed protocols that can compensate for both clock skews and offsets are the Distributed Time-

Sync Protocol in [6] and the Average Time-Sync Protocol in [23]. 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. They both are proved to synchronize a network of clocks in the absence of noise and time-delays and they also show good performance in experimental testbeds. Of note is the fact that these strategies are highly nonlinear and do not lead to a theoretical characterization of the effects of noise on the steady-state performance, i.e., the distribution of synchronization errors.

Differently, [24] 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) consensus based controller that treats the different clock speeds as unknown constant disturbances and the different clock offsets as different initial conditions for the system dynamics. A similar approach, based on the second order consensus algorithm, has been proposed by the same authors in [25] to deal with the synchronization of networks of noisy non-identical double integrators. In [25], both convergence guarantees as well optimal design using standard optimization tools when the underlying communication graph is fixed and known, are provided.

It is important however to remark that the protocols in [24], [25] are proposed in a synchronous implementation, namely, all nodes are assumed to communicate at the same time instants which are multiple integers of a given sampling time T, while the other protocols mentioned above are all asynchronous, i.e., they do not need a specific coordination for the communication. Clearly the assumption made in [24], [25], is unrealistic for implementing any clock synchronization algorithm over WSNs since nodes data transmissions and algorithm updating steps cannot occur exactly at the same time. Indeed, in a fully distributed synchronization algorithm, these operations can be performed by the nodes relying only on their local time estimates and, in absence of synchronization, these estimates differ from each other.

The main contribution of this paper is to propose a realistic implementation of the strategy presented in [25] in the context of WSNs. We refer to this implementation as the *pseudo-synchronous* algorithm. Differently from the synchronous version proposed in [25], in the pseudo-synchronous algorithm each clock is not assumed to communicate with the other clocks synchronously at (absolute) time $t = kT$, $k = 0, 1, 2, \ldots$, but instead it is assumed that it performs the transmission whenever its time estimate is equal to kT. Consequently, in the pseudo-synchronous algorithm, clocks data transmissions associated with the k-th iteration are not performed synchronously. In this paper we show that the pseudo-synchronous algorithm is a perturbation of the synchronous version proposed in [25] and hence, through arguments related to the center manifold theorem, we can prove its convergence under the following assumptions:

- the initial time estimates are sufficiently close to each other (namely we prove only the *local convergence* of the algorithm);
- absence of process noise, measurement noise and propagation delays.

Clearly, from a practical point of view, the above assumptions represent important limitations when dealing with synchronization algorithms running over WSNs. To partially overcome these limitations, we provide a set of numerical simulations showing that the effectiveness of the algorithm is not affected even when the two above assumptions do not hold. In particular, our simulations seem to suggest that the pseudo-synchronous algorithm converges even if the initial time estimates are quite different from each other, namely they suggest that the basin of attraction is large and that it does not depend on the number of clocks.

Additionally, in the section devoted to simulations, we implement an asynchronous version of the PI strategy introduced in [25]. In particular a numerical comparison with the distributed strategies proposed in [6] and in [23], is performed. While being slightly slower to asymptotically converge to the synchronization, the asynchronous version of the PI strategy significantly outperforms the other strategies in terms of robustness to process and measurement noises and time-varying clock drifts.

The outline of the paper is as follows. In Section II the clock model for each clock is presented, together with the formulation of the clock synchronization problem. In Section III the *Pseudo-Synchronous* algorithm is described. Its convergence properties are characterized in Section IV. Numerical examples for comparing our strategy with other fully distributed strategies are provided in Section V. Finally, we gather out our conclusions in Section VI. Moreover, since the center manifold theorem for discrete time systems is not easy to find in the literature, for sake of completeness we prefered to add an appendix in which the version of the center manifold theorem needed in the paper is stated and proved.

A. Mathematical preliminaries

Before proceeding, we collect some useful definitions and notations. In this paper $\mathcal{G} = (V, E)$ denotes an *graph* where $V = \{1, \dots, N\}$ is the set of vertices and $E \subseteq V \times V$ is the set of (directed) edges. The graph $\mathcal{G}$ is said to be *undirected* if $(i, j) \in E$ implies $(j, i) \in E$. A *path* in $\mathcal{G}$ consists of a sequence of vertices $(i_1, i_2, \dots, i_r)$ such that $(i_j, i_{j+1}) \in E$ for every $j \in \{1, \dots, r-1\}$. A graph $\mathcal{G}$ is *strongly connected* if for any pair of vertices (i, j) there exists a path connecting i to j. Given a node i, by $\mathcal{N}_i$ we denote the set of 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 matrix $M \in \mathbb{R}^{N \times N}$, we define the induced graph $\mathcal{G}_M = (V_M, E_M)$ in which $V_M := \{1, \dots, N\}$ and in which $(j, i) \in E_M$ if $M_{ij} \neq 0$. Given a graph

$\mathcal{G} = (V, E)$, the matrix M is *adapted* to (or *compatible* with) $\mathcal{G}$ if $V = V_M$ and $E \supseteq E_M$. Now we give some notational conventions. Given a vector $v \in \mathbb{R}^N$ and a matrix $M \in \mathbb{R}^{N\times N}$, we let v^T and M^T denote respectively 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 $v = [v_1, \ldots, v_N]^T \in \mathbb{R}^N$, diag$\{v\}$ or diag$\{v_1, \ldots, v_N\}$ mean a diagonal matrix having the components of v as diagonal elements. Moreover, $\|v\|$ denote the Euclidean norm of v. Finally we let $\mathbb{R}$ and $\mathbb{R}_{>0}$ denote, respectively, the set of real numbers and the set of strictly positive real numbers. The set of non-negative natural numbers is denoted by $\mathbb{N}$.

II. The clock synchronization problem

A. The mathematical modeling of a clock

We start by proposing a model of each local clock 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 clock has an oscillator capable to produce an event at time $t(k)$, $k \in \mathbb{N}$. The clock has to use these ticks 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

$$s(t) = \sum_{k=0}^{\infty} \mathbf{1}(t - t(k))$$

where $\mathbf{1}$ is the unit step function, i.e.,

$$\mathbf{1}(t) = \begin{cases} 0 & t \leq 0 \\ 1 & t > 0 \end{cases}$$

In this way the counter output is the step shaped function shown in Figure 1. Notice that the function

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

can be interpreted as the oscillator frequency at time t and that $s(t) \simeq \int_{-\infty}^{t} f(\sigma)d\sigma$. Typically a nominal value $\hat{f}$ of $f(t)$ is known together with a lower bound f_{min} and an upper bound f_{max} such that $f(t) \in [f_{min}, f_{max}]$. From the counter one can build a time estimate $x'(t)$ by letting

$$x'(t) = x'(t_0) + x''(t_0)[s(t) - s(t_0)], \tag{II.2}$$

where $x''(t_0)$ is an estimate of the oscillation period $1/f(t)$ in the period $[t_0, t]$. It is reasonable to initialize $x''(t)$ to $1/\hat{f}$. Both $x'(t)$ and $x''(t)$ 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

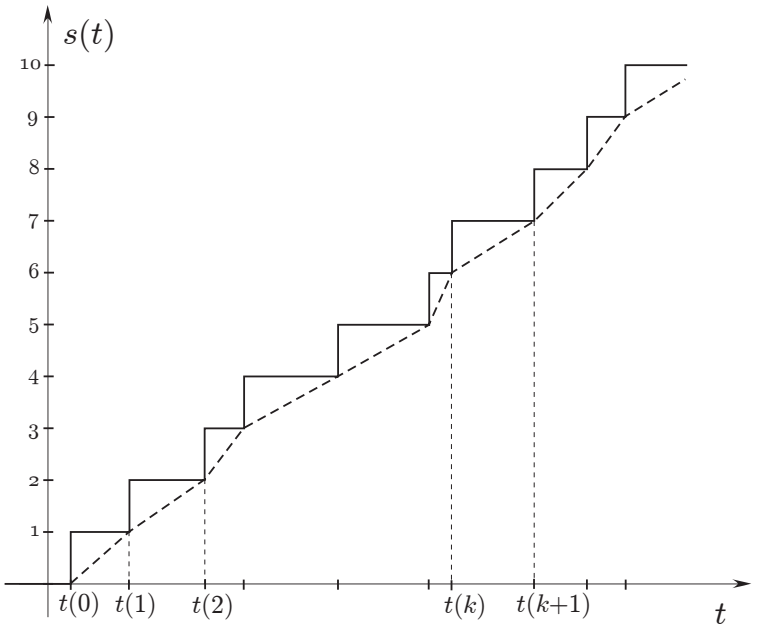

Fig. 1. The graphs of $s(t)$ (continuous line) and of its approximation $\int_{-\infty}^{t} f(\sigma)d\sigma$ (dashed line).

time instants $T_{up}(h)$, $h \in \mathbb{N}$, called updating time instants. In this case we have [1]

$$\begin{cases} x'(T_{up}(h)^+) = x'(T_{up}(h)) + u'(h) \\ \hat{x}''(T_{up}(h)^+) = x''(T_{up}(h)) + u''(h) \end{cases} \tag{II.3}$$

where u' and u'' denote the control inputs applied to x' and x'', respectively.

Clearly, for $t \in (T_{up}(h), T_{up}(h+1))$, the value of x'' is kept constant, i.e, $x''(t) = x''(T_{up}(h)^+)$, while the value of x' is updated according to (II.2)

$$x'(t) = x'(T_{up}(h)^+) + x''(T_{up}(h)^+)[s(t) - s(T_{up}(h))].$$

Now, for $t \in \mathbb{R}$, define the state of the clock to be the two-dimensional real vector $x(t) = [x'(t)\ x''(t)]^T$, $t \in \mathbb{R}$, and define the control input of the clock to be the two-dimensional real vector $u(h) := [u'(h)\ u''(h)]^T$, $h \in \mathbb{N}$. Observe that Equation II.3 can be rewritten as

$$x\left(T_{up}(h)^+\right) = x\left(T_{up}(h)\right) + u(h).$$

[1]Given the time t, with the symbol t^+ we mean the time just after t. More formally we assume that all our functions are simultaneously left continuous and piecewise continuous; for a such given function f, with a slight abuse of notation, with $f(t^+)$ we mean $\lim_{t \to t^+} f(t)$.

Moreover, for $t \in (T_{up}(h), T_{up}(h+1))$, x' is updated according to (II.2), i.e.,

$$x'(t) = x'\left(T_{up}(h)^+\right) + \left(s(t) - s(T_{up}(h))\right) x''\left(T_{up}(h)^+\right)$$

while x'' remains constant, i.e, $x''(t) = x''\left(T_{up}(h)^+\right)$. In compact form, for $t \in (T_{up}(h), T_{up}(h+1))$, we can write

$$x\left(t\right) = \begin{bmatrix} 1 & s(t) - s(T_{up}(h)) \\ 0 & 1 \end{bmatrix} x\left(T_{up}(h)^+\right) \tag{II.4}$$

B. Networked clocks' 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. We assume that the nominal values, the lower and upper bounds $\hat{f}, f_{min}, f_{max}$ are the same for all the clocks. Moreover, for $i \in \{1, \ldots, N\}$, let $x_i(t) = [x_i'(t)\ x_i''(t)]^T$ denote the local state of the clock i. In our model we assume that the clocks can exchange their local state according to a graph $\mathcal{G}$ having $\{1, \ldots, N\}$ as set of vertices and in which there is an edge from j to i whenever the clock j can send its state x_j to the clock i. Specifically, we assume that each clock i broadcasts the value of its state $x_i(t)$ to all its neighboring nodes, at some time instants $T_{tx,i}(h)$, $h \in \mathbb{N}$. In general, due to the transmission delays, the information $x_i(T_{tx,i}(h))$ is received by clock $j \in \mathcal{N}_i$ at a delayed time $T_{rx,i,j}(h)$ which can be conveniently described as

$$T_{rx,i,j}(h) = T_{tx,i}(h) + \gamma_{i,j}(h) \tag{II.5}$$

where $\gamma_{i,j}(h)$ is a nonnegative real number representing the deliver delay between i and j.

Each node i can use any information it receives from the neighboring nodes to perform a control at the time instants $T_{up,i}(h)$, $h \in \mathbb{N}$. The objective is to design a control strategy yielding the clock synchronization, i.e., such that there exist constants $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}$ such that synchronization errors

$$e_i(t) := x_i'(t) - (at + b), \qquad i = 1, \ldots, N \tag{II.6}$$

converge to zero or remain small. In the next section we describe three different examples of synchronization algorithm.

III. The pseudo-synchronous clock synchronization algorithm

Before describing the clock synchronization algorithm we focus in this paper, we present two other possible solutions. The first, even if unrealistic, will be helpful in the presentation of the realistic solution we will propose, the second is instead briefly described for sake of completeness, but will not be analyzed in this paper. In this Section we make the following assumptions.

Assumption 3.1: We assume that the oscillator frequencies are unknown but constant, i.e., for $i \in \{1, \dots, N\}$, $f_i(t) = \bar{f}_i$ for all $t \in \mathbb{R}_{>0}$.

Assumption 3.2: We assume that the delays are negligible, i.e., for $i \in \{1, \dots, N\}$, $T_{rx,i,j}(h) = T_{tx,i}(h)$, for all $j \in \mathcal{N}_i$ and for all $h \in \mathbb{N}$.

We will come back on time-varying oscillator frequencies and on not negligible delays in Section V, devoted to simulations. Observe that, under Assumption 3.1, Equation (II.4) becomes

$$x_i(t) = \begin{bmatrix} 1 & \bar{f}_i(t - T_{up,i}(h)) \\ 0 & 1 \end{bmatrix} x_i\left(T_{up,i}(h)^+\right). \tag{III.1}$$

A. An unrealistic synchronous algorithm

This version is unrealistic, but constitutes the starting point for a more realistic implementation. We first assume that, for all $i \in \{1, \dots, N\}$, $T_{up,i}(h) = T_{tx,i}(h) = hT$ where $T \in \mathbb{R}_{>0}$ is a sampling time, namely, we impose that all the nodes perform their transmissions and the updatings at the same time. Moreover we impose the following linear correction law

$$u_i(h) = -F \sum_{j=1}^{N} K_{ij}[x_j(hT) - x_i(hT)] \tag{III.2}$$

where $F \in \mathbb{R}^{2\times 2}$ and K_{ij} are the elements of a matrix K with $K_{ij} = 0$ if (j, i) is not an edge of the communication graph $\mathcal{G}$. In this way the protocol requires the transmission of the state $x_j(hT)$ from the node j to the node i if and only if $K_{ij} \neq 0$. This implementation is unrealistic, because it is not possible for the nodes to act at the same time instants hT, since they have not a common time coordinate system. If we define the state

$$x(h) := [x_1'(hT), \dots, x_N'(hT), x_1''(hT), \dots, x_N''(hT)]^T \in \mathbb{R}^{2N}$$

it can be verified[2] that this synchronous algorithm evolves according to the following equation

$$x(h+1) = Ax(h) \tag{III.3}$$

where

$$A = \begin{bmatrix} I & TD \\ 0 & I \end{bmatrix} \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} f_{11}K & f_{12}K \\ f_{21}K & f_{22}K \end{bmatrix} \right) \tag{III.4}$$

with f_{ij} being the elements of the matrix F and $D := \text{diag}\{\bar{f}_1, \dots, \bar{f}_N\}$. In case the matrix A yields the clock synchronization, namely the convergence of the N clock profiles $x_i'(hT)$ to a common ramp shaped function, we will call this matrix a *synchronizing* matrix.

[2]The proof of this fact is a simplified version of the proof of Proposition 4.3.

Remark 3.3: The above synchronous algorithm has been analyzed in [26] under the assumption that K is a symmetric matrix such that $K\mathbb{1} = 0$. It has been shown that, in order to obtain a synchronizing matrix A, we need to impose that $f_{22} = 0$. In [26] it is assumed that also $f_{12} = 0$, since this limitation does not prevent from finding a solution and has the advantage that each clock needs to transmit only one real number corresponding to the first component of its state. From [27] it can argued that, given a connected undirected graph $\mathcal{G}$, building the matrix K as follows

$$K_{ij} = \begin{cases} -\frac{1}{\max\{d_i, d_j\}} & \text{if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ -\sum_{j \neq i} K_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{III.5}$$

and choosing $f_{11} = \frac{1}{2}$ and $f_{21} = \frac{1}{f_{max}T}$, we obtain a matrix A which is synchronizing for all oscillator frequencies $\bar{f}_i$, $i \in \{1, \ldots, N\}$ such $0 < \bar{f}_i < f_{max}$. Notice that the proposed construction of K, inspired to the transition matrix of the Metropolis-Hastings algorithm, requires only local information, since every node needs to know only the degree of its neighbor nodes.

B. An asynchronous algorithm

In this algorithm we assume that any node j decides to broadcast at completely random times $T_{tx,j}(h)$ its state $x_j(T_{tx,j}(h))$ to its neighbors. Since we are assuming that there are no delays, any neighbor node i of j receives $x_j(T_{tx,j}(h))$ at the same time $T_{tx,j}(h)$ and instantaneously operates the correction

$$x_i(T_{tx,j}(h)^+) = x_i(T_{tx,j}(h)) - FK_{ij}[x_j(T_{tx,j}(h)) - x_i(T_{tx,j}(h))] \tag{III.6}$$

where $F \in \mathbb{R}^{2\times 2}$ and K_{ij} are the elements of a matrix K. In this way, the updating time instants for the node i coincide with the transmission time instants of all its neighbor nodes.

C. A pseudo-synchronous algorithm

As we noticed above, the synchronous algorithm is unrealistic, since, in any realistic setting, the nodes are unable to carry out their transmissions synchronously. Indeed in a fully distributed synchronization algorithm, these operations can be performed by the nodes relying only on their local time estimates and, in absence of synchronization, these estimates differ from each other. Here we propose a realistic version of the synchronous algorithm. We will call it *pseudo-synchronous* algorithm. In order to describe the pseudo-synchronous algorithm, we first specify how the nodes select the transmission and the updating time instants. Assume that node i, $i \in \{1, \ldots, N\}$, knows its neighbors in the graph $\mathcal{G}$, namely the set $\mathcal{N}_i$, and assume that, as for the synchronous algorithm, we have two matrices matrix $F \in \mathbb{R}^{2\times 2}$ and

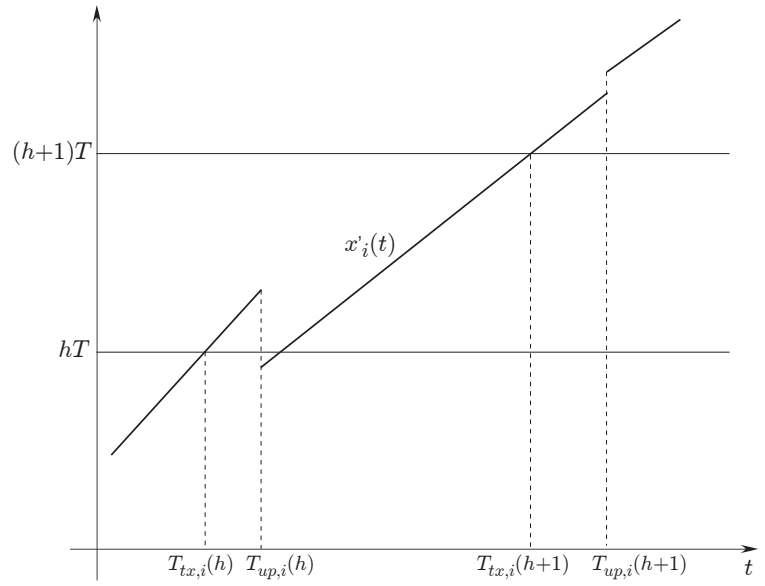

Fig. 2. In this Figure we show how the time instants $T_{tx,i}(h)$ and $T_{up,i}(h)$ are related to the state evolution $x'_i(t)$.

$K \in \mathbb{R}^{N\times N}$. Finally let T be a positive parameter which is *a priori* known by all the clocks. Differently from the synchronous implementation, now T does not denote a sampling interval but just a positive constant common parameter needed to determine when the transmissions have to occur. Then we have the following definitions:

- The transmission time instants are defined by

$$T_{tx,i}(h) := \inf\{t \,:\, x'_i(t) \geq hT\}$$

  namely the i-th node transmits when its time estimate $x'_i(t)$ becomes greater than or equal to hT for the first time. This is an example of an event based decision.

- The updating time instants are defined by

$$T_{up,i}(h) := \max\{\, T_{tx,j}(h) \mid j \in \mathcal{N}_i \cup \{i\}\,\},$$

  namely the node i updates its state only after the completion of all its neighbor nodes communication actions, included its own transmission.

Notice that, from the above definitions, we always have that $T_{up,i}(h) \geq T_{tx,i}(h)$. Moreover, $T_{tx,i}(h)$ and $T_{up,i}(h)$ can be determined by the node i relying only on its local information. Figure 2 depicts the relation between the evolution of the time estimate $x'_i(t)$ and the time instants $T_{tx,i}(h)$ and $T_{up,i}(h)$.

According to the pseudo-synchronous algorithm, during the h-th iteration each node i performs the following actions:

- At instant time $T_{tx,i}(h)$ the node i broadcasts to its neighbors the state $x_i(T_{tx,i}(h))$.
- For all $j \in \mathcal{N}_i$, at time $T_{tx,j}(h)$ the node i receives from the node j the value $x_j(T_{tx,j}(h))$, it computes the difference $x_j(T_{tx,j}(h)) - x_i(T_{tx,j}(h))$ and, it stores it in the memory.
- At time $T_{up,i}(h)$ the node i updates its state using all the stored data as follows

$$x_i(T_{up,i}(h)^+) \;=\; x_i(T_{up,i}(h)) - F \sum_{j\in\mathcal{N}_i} K_{ij}\left[x_j(T_{tx,j}(h)) - x_i(T_{tx,j}(h))\right]. \tag{III.7}$$

Some remarks are now in order.

Remark 3.4: Observe that we could equivalently define $T_{tx,i}(h)$ as follows $T_{tx,i}(h) := \min\{t \,:\, x_i'(t) = hT\}$ with the only additional attention to the case in which it happens that $x_i'(T_{up,i}(h)) < (h+1)T$ and that $x_i'(T_{up,i}(h)^+) > (h+1)T$ since there would exist no time t in which $x_i'(t) = (h+1)T$. Indeed in this case we would need to define $T_{tx,i}(h+1) = T_{up,i}(h)$.

Remark 3.5: There exist scenarios in which the algorithm we proposed does not work. For instance in case the nodes do not know which are their neighbors or in case we might have packet losses. In these situations a node i can not wait until it receives all the data before updating its state. A way to solve this problem is to change the definition of the updating time $T_{up,i}(h)$ as follows

$$T_{up,i}(h) := \inf\{t \,:\, x_i'(t) \geq hT + \epsilon\} \tag{III.8}$$

where ϵ is a constant such that $0 < \epsilon < T$. In this case the correction $u_i(h)$ will be computed in the same way as before, but considering only the information that the node i has received before $T_{up,i}(h)$. Observe that this version of the algorithm makes it suitable also for dealing with cases in which there may be node failures, node insertions and packet losses in the communication network. In this case also the weights K_{ij} need to be changed and adapted to the number of the received information packets. There are various ways of doing this as suggested in [18].

Remark 3.6: Observe that, in case we choose F such that $f_{12} = f_{22} = 0$, then at $T_{tx,i}(h)$ the node i needs to broadcast to its neighbor nodes only the first component $x_i'(T_{tx,i}(h))$ of the state $x_i(T_{tx,i}(h))$ which we know is equal to hT. In other words in this case the information transmitted is essentially only the integer h and not a real number. This is useful in case the transmission is done through a digital channel with low bit rate.

Remark 3.7: Equation (III.7) describing the dynamics of pseudo-synchronous algorithm shows how the states evolve only at times $T_{up,i}(h)$, $h \in \mathbb{N}$, $i \in \{1,\ldots,N\}$. It is worth remarking however that, for $t \in]T_{up,i}(h), T_{up,i}(h+1)[$, the i-th node updates its local state according to (II.4) and (III.1), namely,

$$\begin{aligned} x_i'(t) &= x_i'(T_{up,i}(h)^+) + [s_i(t) - s_i(T_{up,i}(h)^+)]x_i''(T_{up,i}(h)^+) \\ &= x_i'(T_{up,i}(h)^+) + \bar{f}_i(t - T_{up,i}(h)^+)x_i''(T_{up,i}(h)^+) \end{aligned} \tag{III.9}$$

where $s_i(t)$ is the evolution of i-th node counter, and

$$x_i''(t) = x_i''(T_{up,i}(h)^+). \tag{III.10}$$

IV. A DISCRETE TIME NONLINEAR SYSTEM MODEL FOR THE PSEUDO-SYNCHRONOUS ALGORITHM

In this section we discuss the convergence properties of the pseudo-synchronous algorithm and we state the main result of the paper. Before doing so, we first show that the evolution of the pseudo-synchronous algorithm can be modeled by a discrete time nonlinear system that can be written as a perturbation of the discrete time linear system (III.3)- (III.4) describing the evolution of the synchronous algorithm introduced in Section III-A. For simplicity, from now on, we will assume that $f_{12} = f_{22} = 0$, restriction which will not prevent us to find a solution to our problem and that has the advantage that only one component of the state needs to be transmitted. In order to model the pseudo-synchronous algorithm through a discrete time nonlinear system, we need to fix some sampling time instants, since the pseudo-synchronous algorithm is continuous-time, while the synchronous algorithm (III.3) is discrete-time. A suitable definition is given by letting

$$T(h) := \min_i \{T_{tx,i}(h)\}. \tag{IV.1}$$

Loosely speaking $T(h)$ represents the first time instant in which the estimate of a node reaches the value hT. From $T(h)$ we can define the states $x_i'(h) := x_i(T(h))$ and $x_i''(h) := x_i(T(h))$ and the vectors

$$x'(h) := [x_1'(h), \dots, x_N'(h)]^T, \quad x''(h) := [x_1''(h), \dots, x_N''(h)]^T \in \mathbb{R}^N$$

$$x(h) := [x_1'(h), \dots, x_N'(h), x_1''(h), \dots, x_N''(h)]^T \in \mathbb{R}^{2N}.$$

Remark 4.1: Observe that the way we build the discrete time state evolution $x_i'(h)$ and $x_i''(h)$ from the continuous time state evolution $x_i'(t)$ and $x_i''(t)$ is such that state vector $x(h)$ has to belong to a strict subset of $\mathbb{R}^{2N}$. Precisely, we necessary have that $x_i'(h) - hT \leq 0$ for all $i \in \{1, \dots, N\}$ and that there exists $\bar{i} \in \{1, \dots, N\}$ such that $x_{\bar{i}}'(h) = hT$. This holds under the hypothesis that the entries of $x''(t)$ are kept non-negative, which is a reasonable constraint to be imposed to any clock synchronization algorithm which makes sense. Indeed, this last hypothesis simply imposes that the clocks are not allowed "to go back". In fact, since $x_i''(t)$ represents the estimate that i-th clock has of the inverse of its oscillator frequency $\bar{f}_i$, and, since by Assumpion 3.1 the i-th clock knows that for sure these frequency can not be less that f_{min}, then it is reasonable to impose that $x_i''(t) \geq 1/f_{min}$.

Define now

$$\begin{aligned}\delta T(h) &:= T(h+1) - T(h) - T \\ \delta T_{tx,i}(h) &:= T_{tx,i}(h) - T(h) \\ \delta T_{up,i}(h) &:= T_{up,i}(h) - T(h)\end{aligned} \tag{IV.2}$$

and the vectors

$$\delta T_{tx}(h) := [\delta T_{tx,1}(h), \dots, \delta T_{tx,N}(h)]^T \tag{IV.3}$$

$$\delta T_{up}(h) := [\delta T_{up,1}(h), \dots, \delta T_{up,N}(h)]^T . \tag{IV.4}$$

The quantities $\delta T_{up,i}(h)$ and $\delta T_{tx,i}(h)$ accounts for the lack of synchronicity in performing the updating and transmitting actions, while $\delta T(h)$ measures the length of the h-th sample period with the respect to T. The following lemma shows how $\delta T(h)$, $\delta T_{tx}(h)$ and $\delta T_{up}(h)$ depends on the state $x(h)$.

Lemma 4.2: Consider the quantities $\delta T(h)$, $\delta T_{tx}(h)$ and $\delta T_{up}(h)$ defined above. Then:

1) They are functions of h and of the state $x(h)$.
2) They depend continuously on the state $x(h)$.
3) They are zero if

$$x(h) = \begin{bmatrix} hT\mathbb{1} \\ D^{-1}\mathbb{1} \end{bmatrix},$$

where we recall that $D = \text{diag}\{\bar{f}_1, \dots, \bar{f}_N\}$.

Proof:

1,2) From the definition of $\delta T_{tx,i}(h)$ and $\delta T_{up,i}(h)$, $i \in \{1, \dots, N\}$, one can see that

$$\delta T_{tx,i}(h) = T_{tx,i}(h) - T(h) = \frac{hT - x_i'(T(h))}{\bar{f}_i x_i''(T(h))} = \frac{hT - x_i'(h)}{\bar{f}_i x_i''(h)} \tag{IV.5}$$

and that

$$\delta T_{up,i}(h) = \max_{j \text{ s.t. } K_{ij} \neq 0} \{\delta T_{tx,j}(h)\} . \tag{IV.6}$$

From (IV.5) and (IV.6), it directly follows that $\delta T_{tx,i}(h)$, $\delta T_{up,i}(h)$, $i \in \{1, \dots, N\}$, are functions of h and are continuous functions of the state $x(h)$. As far as $\delta T(h)$ is concerned we proceed as follows. Observe first that

$$\begin{aligned}\delta T(h) &= \min_i \Big\{ T_{tx,i}(h+1) \Big\} - T(h) - T = \min_i \Big\{ T_{tx,i}(h+1) - T(h) \Big\} - T \\ &= \min_i \Big\{ T_{tx,i}(h+1) - T_{up,i}(h) + T_{up,i}(h) - T(h) \Big\} - T \\ &= \min_i \Big\{ T_{tx,i}(h+1) - T_{up,i}(h) + \delta T_{up,i}(h) \Big\} - T.\end{aligned}$$

Since we have already proved that $\delta T_{up,i}(h)$ depends on h and depends continuously on $x(h)$, it remains to prove that the same thing holds for the quantity $T_{tx,i}(h+1) - T_{up,i}(h)$. In fact observe that

$$T_{tx,i}(h+1) - T_{up,i}(h) = \frac{(h+1)T - x_i'(T_{up,i}(h)^+)}{\bar{f}_i x_i''(T_{up,i}(h)^+)}. \tag{IV.7}$$

On the other hand we have that

$$x_i''(T_{up,i}(h)^+) = x_i''(T(h)) - f_{21} \sum_{j \in \mathcal{N}_i} K_{ij} \left[hT - x_i'(T_{tx,j}(h)) \right] = x_i''(h) + f_{21} \sum_{j \in \mathcal{N}_i} K_{ij} x_i'(T_{tx,j}(h)) \tag{IV.8}$$

$$x_i'(T_{up,i}(h)^+) = x_i'(T_{up,i}(h)) - f_{11} \sum_{j \in \mathcal{N}_i} K_{ij} \left[hT - x_i'(T_{tx,j}(h)) \right] = x_i'(T_{up,i}(h)) + f_{11} \sum_{j \in \mathcal{N}_i} K_{ij} x_i'(T_{tx,j}(h)) \tag{IV.9}$$

where we used the fact that $\sum_{j \in \mathcal{N}_i} K_{ij} = 0$. Now notice finally that

$$x_i'(T_{tx,j}(h)) = x_i'(T(h)) + \bar{f}_i \, \delta T_{tx,j}(h) \, x_i''(T(h)) = x_i'(h) + \bar{f}_i \, \delta T_{tx,j}(h) \, x_i''(h)$$

$$x_i'(T_{up,i}(h)) = x_i'(T(h)) + \bar{f}_i \, \delta T_{up,i}(h) \, x_i''(T(h)) = x_i'(h) + \bar{f}_i \, \delta T_{up,i}(h) \, x_i''(h)$$

Substituting the previous formulas in (IV.7) and considering that we have already proved that $\delta T_{tx,i}(h), \delta T_{up,j}(h)$, $i \in \{1, \dots, N\}$, are functions of h and are continuous functions of $x(h)$, we obtain that $T_{tx,i}(h+1) - T_{up,i}(h)$ is a function of h and is continuous function of $x(h)$, which yields the assertion.

3) One can see that from (IV.5) it follows that, if $x'(h) = hT\mathbb{1}$, then $\delta T_{tx}(h) = 0$. Moreover, if $\delta T_{tx}(h) = 0$, from (IV.6) we have also that $\delta T_{up}(h) = 0$. Now observe that, if $\delta T_{tx}(h) = \delta T_{up}(h) = 0$ then $T_{up,i}(h) = T(h)$ for all i. Therefore, from (IV.9) and (IV.8) we can argue that $x'(T_{up,i}(h)^+) = x'(T_{up,i}(h)) = x'(T(h)) = hT$ and that $x''(T_{up,i}(h)^+) = x''(T_{up,i}(h)) = x''(T(h)) = \bar{f}_i^{-1}$. On the other hand from the previous equalities and from (IV.7) we get $T_{tx,i}(h+1) - T_{up,i}(h) = T$ and so $\delta T(h) = 0$.

■

The following proposition shows how the pseudo-synchronous algorithm can be written as a discrete time nonlinear system that is a perturbation of linear system (III.3). Now for any $v \in \mathbb{R}^N$ define

$$M(v) := K \text{diag}\{v\} - \text{diag}\{K\, v\}.$$

Proposition 4.3: The state $x(h)$ of the pseudo-synchronous algorithm evolves according to the following equations

$$x(h+1) = Ax(h) + \Phi\Big(\delta T(h), \delta T_{tx}(h), \delta T_{up}(h)\Big) x(h) \tag{IV.10}$$

where

$$A = \begin{bmatrix} I & TD \\ 0 & I \end{bmatrix} \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} f_{11} K & 0 \\ f_{21} K & 0 \end{bmatrix} \right) \tag{IV.11}$$

and

$$\Phi\Big(\delta T(h), \delta T_{tx}(h), \delta T_{up}(h)\Big) = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix}$$

with

$$\begin{aligned}
\Phi_{11} &= f_{21}\Big(\text{diag}\{\delta T_{up}(h)\} - \delta T(h)\ I\Big)DK \\
\Phi_{12} &= \delta T(h)D + \Big(- f_{11}I - f_{21}(T + \delta T(h))D + f_{21}\text{diag}\{\delta T_{up}(h)\}D\Big)M(\delta T_{tx}(h))D \\
\Phi_{21} &= 0 \\
\Phi_{22} &= -f_{21}M(\delta T_{tx}(h))D
\end{aligned}$$

Proof: Observe that, according to the definitions given (IV.1) and in (IV.3), we have that $T_{up,i}(h) \leq T(h+1) \leq T_{up,i}(h+1)$. Therefore, from (III.9) it follows that

$$x_i'(h+1) = x_i'(T_{up,i}(h)^+) + \bar{f}_i\left(T(h+1) - T_{up,i}(h)\right) x_i''(T_{up,i}(h)^+) \tag{IV.12}$$

Consider the term $x_i'(T_{up,i}(h)^+)$. From Equation (III.7) it follows

$$x_i'(T_{up,i}(h)^+) = x_i'(T_{up,i}(h)) - f_{11}\sum_{j\in\mathcal{N}_i} K_{ij}\left(x_j'(T_{tx,j}(h)) - x_i'(T_{tx,j}(h))\right)$$

and hence

$$x_i'(h+1) = x_i'(T_{up,i}(h)) - f_{11}\sum_{j\in\mathcal{N}_i} K_{ij}\left(x_j'(T_{tx,j}(h)) - x_i'(T_{tx,j}(h))\right) + \bar{f}_i\left(T(h+1) - T_{up,i}(h)\right) x_i''(T_{up,i}(h)^+). \tag{IV.13}$$

Now notice that, for $i \in \{1, \ldots, N\}$, $T(h) \leq T_{up,i}(h) \leq T(h+1)$, $T(h) \leq T_{tx,i}(h) \leq T(h+1)$ and $T_{tx,j}(h) \leq T_{up,i}(h)$ for all $j \in \mathcal{N}_i \cup \{i\}$. Hence, from (III.10) we can deduce that

$$\begin{aligned}
x_i''(T_{up,i}(h)) &= x_i''(h) \\
x_i''(T_{up,i}(h)^+) &= x_i''(h+1) \\
x_i''(T_{tx,j}(h)) &= x_i''(h), \qquad \forall j \in \mathcal{N}_i \cup \{i\},
\end{aligned} \tag{IV.14}$$

and that

$$\begin{aligned}
x_i'(T_{up,i}(h)) &= x_i'(h) + \bar{f}_i\left(T_{up,i}(h) - T(h)\right) x_i''(T_{up,i}(h)) = x_i'(h) + \bar{f}_i\delta T_{up,i}(h) x_i''(h) \\
x_j'(T_{tx,j}(h)) &= x_j'(h) + \bar{f}_j\left(T_{tx,j}(h) - T(h)\right) x_j''(T_{tx,j}(h)) = x_j'(h) + \bar{f}_j\delta T_{tx,j}(h) x_j''(h) \\
x_i'(T_{tx,j}(h)) &= x_i'(h) + \bar{f}_i\left(T_{tx,j}(h) - T(h)\right) x_i''(T_{tx,j}(h)) = x_i'(h) + \bar{f}_i\delta T_{tx,j}(h) x_i''(h)
\end{aligned} \tag{IV.15}$$

Moreover we can write

$$\bar{f}_i \left(T(h+1) - T_{up,i}(h)\right) x_i''(T_{up,i}(h)^+) = \bar{f}_i \left(T(h+1) - T(h) - T + T + T(h) - T_{up,i}(h)\right) x_i''(h+1)$$
$$= \bar{f}_i T x_i''(h+1) + \bar{f}_i \delta T(h) x_i''(h+1) - \bar{f}_i \delta T_{up,i}(h) x_i''(h+1) \tag{IV.16}$$

where in the last equality we used the definition of $\delta T(h)$ provided in (IV.3). Plugging expressions given in (IV.15) and in (IV.16) into Equation (IV.13), we get

$$\begin{aligned} x_i'(h+1) = x_i'(h) + \bar{f}_i T x_i''(h+1) - f_{11} \sum_{j \in \mathcal{N}_i} K_{ij} \left(x_j'(h) - x_i'(h)\right) + \\ - f_{11} \sum_{j \in \mathcal{N}_i} K_{ij} \delta T_{tx,j}(h) \left(\bar{f}_j x_j''(h) - \bar{f}_i x_i''(h)\right) + \bar{f}_i \delta T(h) x_i''(h+1) + \\ + \bar{f}_i \delta T_{up,i}(h) \left(x_i''(h) - x_i''(h+1)\right) \end{aligned} \tag{IV.17}$$

As far as the variable x_i'' is concerned, observe that, from Equation (III.7) we can write that

$$\begin{aligned} x_i''(h+1) &= x_i''(T_{up,i}(h)^+) \\ &= x_i''(T_{up,i}(h)) - f_{21} \sum_{j \in \mathcal{N}_i} K_{ij} \left(x_j'(T_{tx,j}(h)) - x_i'(T_{tx,j}(h))\right) \end{aligned}$$

Plugging expressions given in (IV.14) and (IV.15) into the above equation, we get that

$$\begin{aligned} x_i''(h+1) = x_i''(h) - f_{21} \sum_{j \in \mathcal{N}_i} K_{ij} \left(x_j'(h) - x_i'(h)\right) + \\ - f_{21} \sum_{j \in \mathcal{N}_i} K_{ij} \delta T_{tx,j}(h) \left(\bar{f}_j x_j''(h) - \bar{f}_i x_i''(h)\right) \end{aligned} \tag{IV.18}$$

Observe now that the term $\sum_{j \in \mathcal{N}_i} K_{ij}\ \delta T_{tx,j}(h) \left(\bar{f}_j x_j''(h) - \bar{f}_i x_i''(h)\right)$ is the i-th entry of the vector $M(\delta T_{tx}(h)) D x''(h)$, the term $\bar{f}_i \delta T(h) x_i''(h+1)$ is the i-th entry of the vector $\delta T(h) D x''(h+1)$ and the term $\bar{f}_i \left(T_{up,i}(h) - T(h)\right) \left(x_i''(h) - x_i''(h+1)\right)$ is the i-th entry of the vector $\text{diag}\{\delta T_{up}(h)\} D \left(x''(h) - x''(h+1)\right)$. It follows that the N equations in (IV.17) can be collected in

$$\begin{aligned} x'(h+1) =& x'(h) + T D x''(h+1) - f_{11} K x'(h) - f_{11} M(\delta T_{tx}(h)) D x''(h) \\ &+ \delta T(h) D x''(h+1) + \text{diag}\{\delta T_{up}(h)\} D \left(x''(h) - x''(h+1)\right) \end{aligned} \tag{IV.19}$$

while the N equation in (IV.18) can be collected in

$$x''(h+1) = x''(h) - f_{21} K x'(h) - f_{21} M(\delta T_{tx}(h)) D x''(h). \tag{IV.20}$$

Plugging (IV.20) into (IV.19) and rearranging the equations we get the conclusion.

■

Observe that the above system can be written as the sum of the synchronous system (III.3) and some perturbative terms which arise since the clocks do not carry out synchronously their transmitting and updating actions. The following result characterizes the asymptotic synchronization of the pseudo-synchronous algorithm in terms of synchronization error $e(h)$ defined as

$$e(h) = x'(h) - \frac{1}{N}\left(\mathbb{1}^T x'(h)\right)\mathbb{1}.$$

Theorem 4.4: Consider the system (IV.10) and assume that the design parameters f_{11}, f_{21} and K are such that the matrix A defined in (IV.11) is synchronizing according to the notion given in subsection III-A. Then, for all $\epsilon > 0$, there exist neighborhoods $W', W'' \subseteq \mathbb{R}^N$ of the origin such that, if

$$\begin{array}{rcll} x'(0) & \in & \alpha\mathbb{1} + W' & \exists \alpha \in \mathbb{R} \\ x''(0) & \in & D^{-1}\mathbb{1} + W'' & \end{array} \tag{IV.21}$$

then the following properties hold true:

1) $\lim_{h\to\infty}(T(h+1) - T(h)) = \bar{T}$ where $|\bar{T} - T| \leq \epsilon$.
2) The synchronization error $e(h)$ converges exponentially fast to 0, i.e., there exists $C > 0$ and $0 \leq \rho < 1$ such that

$$||e(h)|| \leq C\rho^h ||e(0)||.$$

Proof: In order to proof the theorem we need to find a suitable state transformation able to translate the synchronization into the stability of a suitable nonlinear discrete time system. Then the center manifold theorem will be applied. First consider any $N \times N$ orthonormal matrix matrix having $N^{-1/2}\mathbb{1}^T$ as the first row and let $U \in \mathbb{R}^{N-1\times N}$ the matrix formed by the last $N-1$ rows of this matrix. Let moreover w be a nonzero vector such that $w^T K = 0$. Notice that we have $U\mathbb{1} = 0, \qquad UU^T = I$. Notice moreover that, since $K\mathbb{1} = 0$ and $M(v)\mathbb{1} = 0$, than we have that $K = KU^T U$ and $M(v) = M(v)U^T U$. Define the variables $\bar{x}'(h) := Ux'(h)$, $\bar{x}''(h) := UDx''(h)(h)$ and $r(h) = w^T x''(h) - w^T D^{-1}\mathbb{1}$. Introduce finally the vector $\bar{x}(h) := [\bar{x}'(h)^T \ \bar{x}''(h)^T]^T$. From these definitions and from (IV.10) we can argue that

$$\bar{x}''(h+1) = \bar{x}''(h) - f_{21}UDKU^T\bar{x}'(h) - f_{21}UDM(\delta T_{tx}(h))U^T\bar{x}''(h)$$

and that

$$r(h+1) = r(h) - f_{21}w^T M(\delta T_{tx}(h))U^T\bar{x}''(h).$$

Moreover, from some simple computations we can obtain moreover that

$$\begin{aligned}\bar{x}'(h+1) = &[I - f_{21}TUDKU^T - f_{11}UKU^T]\bar{x}'(h) + T\bar{x}''(h)\\ &+ [f_{21}U\text{diag}\{\delta T_{up}(h)\}DKU^T - f_{21}\delta T(h)UDKU^T]\bar{x}'(h)\\ &+ [-f_{21}TUDM(\delta T_{tx}(h))U^T + f_{21}U\text{diag}\{\delta T_{up}(h)\}DM(\delta T_{tx}(h))U^T\\ &- f_{11}UM(\delta T_{tx}(h))U^T + \delta T(h)I - f_{21}\delta T(h)UDM(\delta T_{tx}(h))U^T]\bar{x}''(h)\end{aligned}$$

Summarizing, we have that

$$\begin{aligned}\bar{x}(h+1) &= \bar{A}\bar{x}(h) + \bar{\Phi}(\delta T_{tx}(h), \delta T_{up}(h), \delta T(h))\bar{x}(h)\\ r(h+1) &= r(h) + \eta(\delta T_{tx}(h), \delta T_{up}(h), \delta T(h))\bar{x}(h)\end{aligned}$$

where

$$\bar{A} = \begin{bmatrix} I - f_{11}UKU^T - f_{21}TUDKU^T & TI \\ -f_{21}UDKU^T & I \end{bmatrix}$$

and where

$$\bar{\Phi}\Big(\delta T(h), \delta T_{tx}(h), \delta T_{up}(h)\Big) = \begin{bmatrix} \bar{\Phi}_{11} & \bar{\Phi}_{12} \\ \bar{\Phi}_{21} & \bar{\Phi}_{22} \end{bmatrix}, \qquad \eta\Big(\delta T(h), \delta T_{tx}(h), \delta T_{up}(h)\Big) = \begin{bmatrix} \eta_1 & \eta_2 \end{bmatrix}$$

with

$$\begin{aligned}\eta_1 &= 0\\ \eta_2 &= -f_{21}w^TM(\delta T_{tx}(h))U^T\end{aligned}$$

and

$$\begin{aligned}\bar{\Phi}_{11} &= f_{21}U\text{diag}\{\delta T_{up}(h)\}DKU^T - f_{21}\delta T(h)UDKU^T\\ \bar{\Phi}_{12} &= -f_{21}TUDM(\delta T_{tx}(h))U^T + f_{21}U\text{diag}\{\delta T_{up}(h)\}DM(\delta T_{tx}(h))U^T\\ &\quad -f_{11}UM(\delta T_{tx}(h))U^T + \delta T(h)I - f_{21}\delta T(h)UDM(\delta T_{tx}(h))U^T\\ \bar{\Phi}_{21} &= 0\\ \bar{\Phi}_{22} &= -f_{21}UDM(\delta T_{tx}(h))U^T\end{aligned}$$

Notice that $\bar{\Phi}$ and η are continuous functions of $\delta T_{tx}(h)$, $\delta T_{up}(h)$ and $\delta T(h)$, which, by Lemma 4.2, are continuous functions of $x(h)$. Finally, taking into consideration Remark 4.1, we can argue that $x(h)$ is a continuous function of $\bar{x}(h),\ r(h)$. Therefore we can conclude that $\bar{\Phi}$ and η are continuous functions of

$\bar{x}(h),\ r(h)$. On the other hand, $\bar{x}(h) = 0,\ r(h) = 0$ implies that $x'(h) = \alpha\mathbb{1}$ for some $\alpha \in \mathbb{R}$ and that $x''(h) = D^{-1}\mathbb{1}$, and, by Lemma 4.2, these two facts imply that $\delta T_{tx}(h)$, $\delta T_{up}(h)$ and $\delta T(h)$ are zero.

Observe finally that, since

$$\bar{A} = \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix} A \begin{bmatrix} U^T & 0 \\ 0 & U^T \end{bmatrix},$$

then, as shown in [26], $\bar{A}$ is an asymptotically stable matrix if and only if the matrix A is synchronizing. Finally notice that, (IV.21) implies that $\bar{x}(0)$ and $r(0)$ belong to a suitably small neighbor of the origin. We are therefore in a position to apply Theorem A.1 in the appendix, which is a simplified version of the center manifold theorem, obtaining in this way the thesis. ■

Remark 4.5: We underline that the previous theorem provides only a local stability result, namely, it only ensures that clocks will reach asymptotic synchronization if their initial states are not too far apart. Moreover, we do not have theoretical results about the behavior of the proposed algorithm with respect to delays/noises. Observe that the iteration of the algorithm is described by a nonlinear discrete-time system and that the main difficulty is that we don't have to prove the stability of this system but only the synchronization. The first part of the paper was to translate the original synchronization problem into a stability problem of a reduced order nonlinear system. To obtain stability results for nonlinear system there are essentially three methods: linearization, center manifold and Lyapunov function. Linearization, which does not work in our case, and center manifold theorem allow to prove only local stability. In order to prove global stability, or to obtain estimates of the basin of attraction, we would need to find a Lyapunov function for our nonlinear system. A Lyapunov function would permit also to obtain a theoretical analysis of the algorithm in the presence of delays/noises by using input to state stability techniques. Unfortunately, we have not been able to find a Lyapunov function for our nonlinear system so far. However, though the convergence is theoretically guaranteed only if the initial states are not too far apart, simulations seem to suggest that the pseudo-synchronous algorithm converges for initial conditions belonging to quite large neighborhoods whose sizes, in general, do not go to zero as N increases.

Remark 4.6: Concerning the parameter ρ, our numerical simulations suggest that it is drastically affected by the communication topology $\mathcal{G}$ and, in turn, by the matrix K. In particular assume that K is built according to the Metropolis weights and let ρ_{ess} the essential spectral radius of the matrix $I - K$, namely, the absolute value of the largest eigenvalue different from 1 of the matrix K. It is well known that ρ_{ess} measures the speed of convergence of the standard consensus algorithm adopting the doubly stochastic matrix $I - K$. It has been proved in the literature that, for many families of graphs, the value of ρ_{ess} goes to 1 as N goes to ∞, namely, the larger the graph is, the slower the corresponding

consensus algorithm is. In our simulations we have noticed that, as ρ_{ess} goes to 1, also the value of the parameter ρ in Theorem 4.4 goes to 1. It will be our future work to provide an upper bound on ρ, eventually depending on the value of the essential spectral radius of the matrix $I - K$.

V. Numerical Examples

A. Implementation examples of the pseudo-synchronous algorithm

In this section we provide two examples illustrating the approach proposed in this paper. Specifically, in Example 5.1 we simulate the pseudo-synchronous algorithm starting from different initial conditions, in Example 5.2 we simulate the effects of the communication delays and packet losses.

Example 5.1: In this simulation we consider a strongly connected random geometric graph with $N = 50$ points uniformly distributed in the unit square, and then placing an edge between each pair of points at distance less than 0.4. We simulate the behavior of the *pseudo-synchronization* algorithm illustrated in Section III-C. We assume that $\bar{f}_i$ are uniformly chosen in $[f_{\min}, f_{max}]$ and we distinguish three cases

1) $f_{min} = 1 - 10^{-1},\ f_{max} = 1 + 10^{-1}$;
2) $f_{min} = 1 - 10^{-2},\ f_{max} = 1 + 10^{-2}$;
3) $f_{min} = 1 - 10^{-3},\ f_{max} = 1 + 10^{-3}$;

In all the three cases we assume that $\hat{f} = 1$. In the synchronization algorithm we assume that $T = 100$ s, and we choose the algorithm parameter according to what suggested in Remark 3.3, namely we choose $f_{11} = 1/2,\ f_{21} = 1/(f_{max}T)$ and the matrix K built according to (III.5). The results obtained are reported in Figure 3 where we plot the trajectories of the quantity $\log N^{-1/2}\|e(h)\|$, generated starting from different initial conditions. Precisely, in the first case $x_i'(0)$ is uniformly distributed in $[0, 10]$, in the second case $x_i'(0)$ is uniformly distributed in $[0, 1]$, in the third case $x_i'(0)$ is uniformly distributed in $[0, 10^{-1}]$, while in all cases we assume that $x_i''(0) = \hat{f} = 1$. The plots reported are the result of the average over 1000 Monte Carlo runs, randomized with respect to both the graph and the initial conditions. The dashed blue line refers to the initial conditions $\bar{f}_i \in [0.9, 1.1]$, $x_i'(0) \in [0, 10]$, the solid red line to $\bar{f}_i \in [0.99, 1.01]$, $x_i'(0) \in [0, 1]$ and the dashed black line to $\bar{f}_i \in [0.999, 1.001]$, $x_i'(0) \in [0, 0.1]$. As expected the smaller the uncertainties on the initial conditions are, the better the performance are, even though, all the trajectories seem to converge exponentially to zero with the same rate of convergence.

Example 5.2: In this example we deal with two practical limitations that must be taken into account when implementing the pseudo-synchronous algorithm over WSNs, namely, not negligible transmission delays and packet losses. We recall that, for $h \in \mathbb{N}$, we denote the deliver delay between j and i as $\gamma_{i,j}(h)$, i.e., $T_{rx,i,j}(h) = T_{tx,i}(h) + \gamma_{i,j}(h)$. Since the i-th node receives the information sent by the j-th node

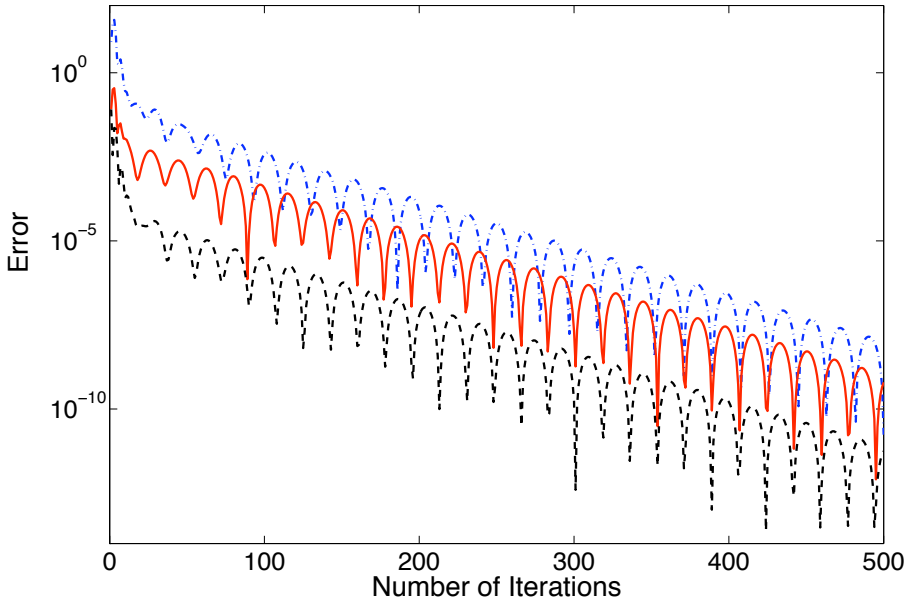

Fig. 3. Trajectories of the synchronization error generated by the pseudo-synchronous algorithm starting from different initial conditions. The dashed blue line refers to the initial conditions $\bar{f}_i \in [0.9, 1.1]$, $x'_i(0) \in [0, 10]$, the solid red line to $\bar{f}_i \in [0.99, 1.01]$, $x'_i(0) \in [0, 1]$ and the dashed black line to $\bar{f}_i \in [0.999, 1.001]$, $x'_i(0) \in [0, 0.1]$.

at time $T_{tx,j}(h)$, at the delayed time $T_{rx,j,i}(h)$ we have that the difference $x'_j\left(T_{tx,j}(h)\right) - x'_i\left(T_{tx,j}(h)\right)$ used in (III.7) must be replaced by $x'_j\left(T_{tx,j}(h)\right) - x'_i\left(T_{rx,j,i}(h)\right)$. Accordingly the input applied by the i-th node becomes

$$u_i(T_{up,i}(h)) = \begin{bmatrix} f_{11} \\ f_{12} \end{bmatrix} \sum_{j \in N_i} K_{ij} \left(x'_j\left(T_{tx,j}(h)\right) - x'_i\left(T_{rx,j,i}(h)\right) \right). \tag{V.1}$$

However, if we model $\gamma_{i,j}(h)$ as a nonnegative random variable of mean $\bar{\gamma}$ and variance σ_γ and if we assume that the value of $\bar{\gamma}$ is known to the clocks[3], we could modify the above equation adding to the term $x'_j\left(T_{tx,j}(h)\right) - x'_i\left(T_{rx,j,i}(h)\right)$ a correcting term trying to compensate the effects due to the delivery delay. More precisely, observe that

$$x'_i\left(T_{tx,j}(h)\right) = x'_i\left(T_{rx,j,i}(h) - \gamma_{j,i}(h)\right) = x'_i\left(T_{rx,j,i}(h)\right) - \gamma_{j,i}(h)\bar{f}_i x''_i\left(T_{rx,j,i}(h)\right),$$

where the unknown quantities $\gamma_{j,i}(h)$ and $\bar{f}_i$ can be suitably approximated by $\gamma_{j,i}(h) \approx \bar{\gamma}$ and $\bar{f}_i \approx \hat{f}$.

[3]The information about $\bar{\gamma}$ might be available from a-priori statistical studies as suggested in [28].

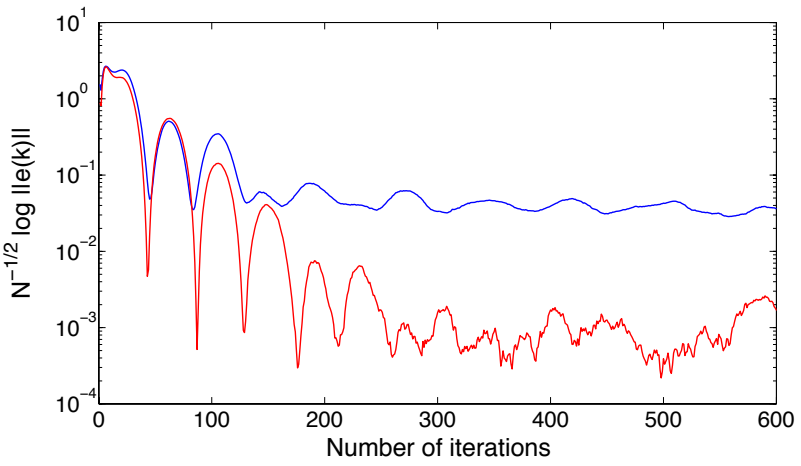

Fig. 4. Effects of the communication delays and packet losses on the performance of the pseudo-synchronous algorithm.

In this way we obtain $x_i'\left(T_{tx,j}(h)\right) \approx x_i'\left(T_{rx,j,i}(h)\right) - \bar{\gamma}\hat{f}x_i''\left(T_{rx,j,i}(h)\right)$ and, thereby,

$$u_i(T_{up,i}(h)) = \begin{bmatrix} f_{11} \\ f_{12} \end{bmatrix} \sum_{j \in N_i} K_{ij} \left(x_j'\left(T_{tx,j}(h)\right) - x_i'\left(T_{rx,j,i}(h)\right)\right) + \bar{\gamma}\hat{f}x_i''\left(T_{rx,j,i}(h)\right). \tag{V.2}$$

As far as packet losses are concerned, we adopt the the modified version of the pseudo-synchronous algorithm proposed in Remark 3.5 where, for $h \in \mathbb{N}$ and for $i \in \{1, \ldots, N\}$, the updating time $T_{up,i}(h)$ is defined as in (III.8). Let $\mathcal{N}_i'$ represent the subset of nodes from which node i has received information before the time instant $T_{up,i}(h)$, i.e., $\mathcal{N}_i' = \{j \in \mathcal{N}_i \,|\, T_{rx,j,i}(h) \leq T_{up,i}(h)\}$, then, at time instant $T_{up,i}(h)$, node i will apply the input

$$u_i(h) = \begin{bmatrix} f_{11} \\ f_{12} \end{bmatrix} \sum_{j \in \mathcal{N}_i'} \frac{1}{|\mathcal{N}_i'| + 1} \left(x_j'\left(T_{tx,j}(h)\right) - x_i'\left(T_{rx,j,i}(h)\right)\right), \tag{V.3}$$

or, if the correction term is applied, the input

$$u_i(h) = \begin{bmatrix} f_{11} \\ f_{12} \end{bmatrix} \sum_{j \in \mathcal{N}_i'} \frac{1}{|\mathcal{N}_i'| + 1} \left(x_j'\left(T_{tx,j}(h)\right) - x_i'\left(T_{rx,j,i}(h)\right)\right) + \bar{\gamma}\hat{f}x_i''\left(T_{rx,j,i}(h)\right). \tag{V.4}$$

In this example, we assume that the graph G is generated as in the previous Example. We assume that $\bar{f}_i$ are uniformly distributed in $[f_{min}, f_{max}]$ where $f_{min} = 1 - 10^{-1}$, $f_{max} = 1 + 10^{-1}$. Again $T = 100$, $f_{11} = 1/2$, $f_{21} = 1/(2Tf_{max})$. Moreover we set $\epsilon = 10$. The initial conditions $x_i'(0)$ are uniformly distributed in $[0, 5]$ and $x_i''(0) = \hat{f} = 1$. The plot reported is the result of the average over 1000 Monte Carlo runs, randomized with respect to both the graph and the initial conditions.

We assume that the delay $\gamma_{i,j}$, for all $(i, j) \in G$ and for all $h \in \mathbb{N}$, is uniformly distributed in the interval $[0, 1]$. Moreover, we assume that any transmitted packet is correctly received with a probability

$p = 0.8$. We implement the pseudo-synchronous both without delay-compensation (blue line) and with delay-compensation (red line) and the results obtained are reported in Figure 4 where we depicted the behavior of the quantity $\log N^{-1/2}\|e(h)\|$. Clearly the presence of the delays and packet losses prevents our algorithm to reach the asymptotic synchronization. However observe that, at the steady-state the variable $\log N^{-1/2}\|e(h)\|$ oscillates within an interval whose amplitude is smaller that 10^{-1}, i.e., one order of magnitude less than the maximum value that the delays can assume. Finally we can see that the correcting term introduced in (V.2) provides a significant improvement of the attainable performance.

B. An asynchronous implementation

The goal of this subsection is to show the effectiveness of the approach proposed in this paper also in the asynchronous scenario that was shortly described in Section III-B. To do so, we propose a comparison between a randomized implementation of (III.6) and the *Average TimeSync* algorithm (denoted hereafter with the shorthand ATS), introduced in [23]. It is worth remarking that the ATS algorithm is a fully distributed asynchronous algorithm, namely, it does not need any specific coordination among the nodes for the communication. The algorithm is based on the cascade of two first order consensus algorithms and it is shown to be provably convergent under the assumption of absence of process noise, measurement noise and propagation delays. We proceed now by describing in more details the control law proposed in (III.6). Let $\mathcal{G} = (V, \mathcal{E})$ be a generic undirected graph. Recall that, for $j \in V$, $\mathcal{N}_j$ denotes the set of the neighbors of j in the graph $\mathcal{G}$. At time $T_{tx,j}(h)$, node j performs its h-th transmission broadcasting the estimate $x'_j\left(T_{tx,j}(h)\right)$ to all its neighbors in the graph $\mathcal{G}$. Under the assumption of no transmission delays and no packet losses, we have that, for $i \in \mathcal{N}_j$, node i receives the information $x'_j\left(T_{tx,j}(h)\right)$ exactly at time $T_{tx,j}(h)$ and, instantaneously, it applies the correction

$$u_i(T_{tx,j}(h)) = \begin{bmatrix} f_{11} \\ f_{21} \end{bmatrix} \left(x'_j(T_{tx,j})(h) - x'_i(T_{tx,j})(h)\right)$$

which yields the updating step

$$x_i(T_{tx,j}(h)^+) = x_i(T_{tx,j}(h)) + \begin{bmatrix} f_{11} \\ f_{21} \end{bmatrix} \left(x'_j(T_{tx,j})(h) - x'_i(T_{tx,j})(h)\right). \tag{V.5}$$

Clearly, for all nodes ℓ which are not neighbors of node i, i.e., $\ell \notin \mathcal{N}_i$, we have $u_\ell(T_{tx,j}(h)) = 0$ and, in turn, $x_\ell(T_{tx,j}(h)^+) = x_\ell(T_{tx,j}(h))$. Defining the matrix K_j by

$$K_j := \sum_{i \in \mathcal{N}_j} \left(e_i e_i^T - e_i e_j^T\right). \tag{V.6}$$

we can write, in more compact form, that

$$x(T_{tx,j}(h)^+) = \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} f_{11}K_j & 0 \\ f_{21}K_j & 0 \end{bmatrix} \right) x(T_{tx,j}(h))$$

In the experimental results reported in the following examples, we assume that, for $j \in \{1, \ldots, N\}$, the transmissions time instants $\{T_{tx,j}(h),\, h \in \mathbb{N}\}$, are generated according to a Poisson process of intensity λ. Moreover, we assume that the Poisson processes generating $\{T_{tx,i}(h),\, h \in \mathbb{N}\}$ and $\{T_{tx,j}(h),\, h \in \mathbb{N}\}$ are independent from each other if $i \neq j$.

Example 5.3 (Speed of convergence to synchronization): In this example we provide a comparison between the asynchronous implementation above described and the ATS algorithm introduced in [23], with the respect to the speed of convergence to the synchronization. We consider a undirected communication graph $\mathcal{G}$ which is a strongly connected realization of a two-dimensional random geometric graph, where vertices are 30 points uniformly distributed in the unit square, and there is a pair of edges (i, j) and (j, i) whenever points i, j have a distance smaller than 0.4. In Figure 5, we depict for both strategies the behavior of $\log N^{-1/2}\|e\|$ being e the synchronization error defined as $e = \Omega x'$ where $\Omega = I - \frac{1}{N}\mathbb{1}\mathbb{1}^T$. The dashed red curve refers to the approach proposed in this paper, while the solid blue curve to the ATS algorithm. The two strategies have been simulated using the same transmission times. Moreover the control parameters of both algorithms have been experimentally designed in order to maximize the speed of convergence to the synchronization.

The plots reported have been obtained averaging over 1000 simulations; a different random geometric graph and initial conditions are independently generated for each simulation. From Figure 5, one can see that the ATS algorithm outperforms, with the respect to the speed of convergence, the performance of the asynchronous implementation in (V.5).

Example 5.4 (Robustness with the respect to communication noises and time-varying oscillator frequencies): In this example we assume that

1) the information exchanged by the nodes is affected by communication noise; and
2) the oscillator frequencies defined in (II.1) are time-varying.

Specifically, we assume that if $x'_j(T_{tx,j}(h))$ is any information transmitted by node j to node i at time $T_{tx,j}(h)$, then node i receives the information $x'_j(T_{tx,j}(h)) + n_{j\to i}(h)$ where $n_{j\to i}(h)$ is a white noise of bounded support. As far as the oscillator frequencies are concerned, we assume that they are modeled as saturated random walks, namely, for each $i \in \{1, \ldots, N\}$, the value of f_i is always within the interval $\left[\hat{f} - \epsilon, \hat{f} + \epsilon\right]$ for some ϵ such that $0 < \epsilon < \hat{f}$. In Figure 6, we depict the behavior of $\log N^{-1/2}\|e\|$ being e the synchronization error defined as previously. The control parameters for both

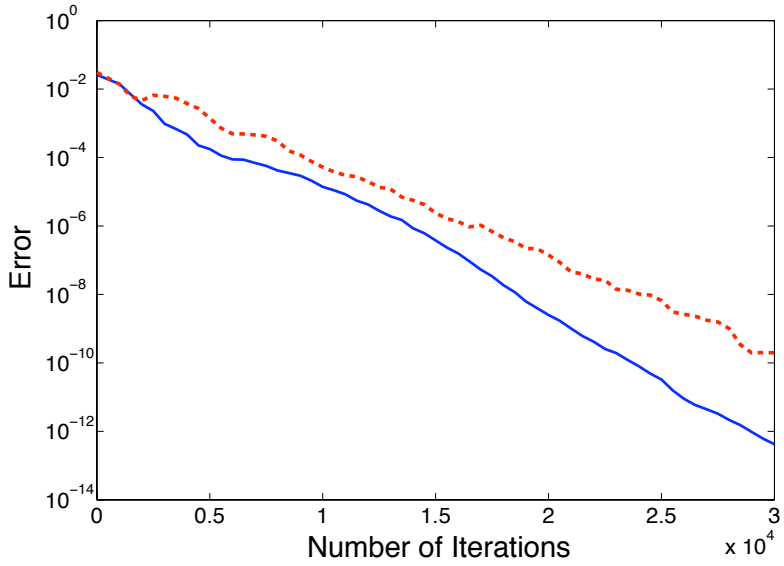

Fig. 5. Comparison in terms of speed of convergence between the algorithm proposed in this paper in its asynchronous implementation (dashed red line) and the ATS algorithm (solid blue line).

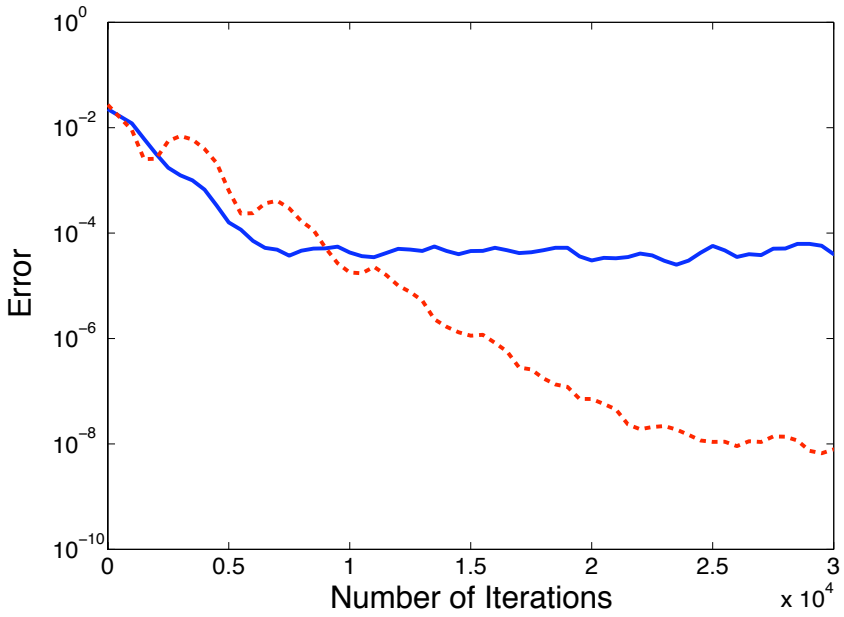

Fig. 6. Comparison in terms of robustness to communication noises and time-varying oscillator frequencies between the algorithm proposed in this paper in its asynchronous implementation (dashed red line) and the ATS algorithm (solid blue line).

algorithms have been chosen experimentally in such a way to minimize the steady state value of $\|e\|$. The plots reported have been obtained averaging over 1000 simulations; a different random geometric graph and initial conditions are independently generated for each simulation. From Figure 6, one can see that the asynchronous implementation described in (V.5), outperforms, with the respect to robustness to communication noises and time-varying oscillator frequencies, the ATS algorithm.

Remark 5.5: Besides the comparison provided in the previous two examples, it is worth stressing also the fact that the ATS algorithm requires, in general, higher computational and memory capabilities than the strategy in (V.5). Indeed, as described in [23], in implementing the ATS algorithm, each node has to perform some updating actions which are nonlinear and has to keep in memory a number of variables which is proportional to the number of its neighbors.

Remark 5.6: The Distributed Time-Sync Protocol is an another fully distributed algorithm recently proposed in the literature to solve the clock synchronization problem, see [6]. The authors of [6], for simplicity, restrict themselves to the case where all the clocks have exactly the same constant oscillator frequency, but have different initial offsets. The offsets compensation is posed as a least-squares problem which is solved in a distributed way through a gradient-based method. The authors mention that a similar least-squares approach could be used also to solve the case of different clocks frequencies. We have run a number of simulations implementing this cascade of two least-squares solvers and we have observed that the performance of the this protocol is comparable with the performance of the ATS algorithm with the respect to both the speed of convergence and the robustness to communication noise.

VI. Conclusions

In this paper we proposed a novel distributed clock synchronization algorithm based on the consensus of higher order linear systems. This algorithm, called *pseudo-synchronous algorithm*, is proved to be a perturbation of a synchronous algorithm which has been recently proposed in the context of synchronization of noisy double integrators but which can not be directly implemented for the clock synchronization. The pseudo-synchronous algorithm is proved to converge under suitable assumptions, namely absence of process noise, measurement noise and propagation delays. Finally, the effectiveness of the algorithm is verified also in presence of time-varying drifts, unreliable communications, packet losses and delivery delays through numerical simulations.

We believe that, in the study of the pseudo-synchronous algorithm, there remain many interesting open issues. Following Remark 4.5, we plan to explore the existence of a Lyapunov function in order to estimate the basin of attraction of the nonlinear system (IV.10) and to perform a theoretical analysis of

the algorithm in presence of delays and noises by using input to state stability techniques. Alternatively, to prove global stability, we plan to investigate the applicability of the contraction theory which has been recently proposed in the study of synchronization of complex systems (see [29]). Additionally we plan to propose variations of the pseudo-synchronous algorithm capable of adapting to time-varying scenarios such as problem in which a node dies or a new node is added to the network. In particular, we envision to endow each node with adaptive laws for the real-time tuning of the parameters f_{11}, f_{12} based only on the local information coming from neighboring nodes. Finally, a challenging open problem is to provide a mathematical proof of the convergence of asynchronous implementation.

APPENDIX

A. The center manifold theorem

In the proof of Theorem 4.4 we need the convergence result stated in Theorem A.1, which is a version of the center manifold theorem for discrete time systems. The center manifold theorem is very rarely presented and almost never proved for discrete time systems. In [30] it is proposed a version in which the functions defining the nonlinear system are assumed to be of a regularity which is actually not satisfied in our case. For this reason, and also for sake of completeness, we propose in this appendix a version of center manifold theorem in which the functions possess the right regularity.

Consider the following discrete time nonlinear system

$$\begin{cases} x(k+1) &= Ax(k) + B(x(k), z(k))x(k) \\ z(k+1) &= z(k) + D(x(k), z(k))x(k) \end{cases} \tag{A.1}$$

where $x(k) \in \mathbb{R}^n$, $z(k) \in \mathbb{R}^m$, $A, B(\cdot,\cdot) \in \mathbb{R}^{n\times n}$ and $D(\cdot,\cdot) \in \mathbb{R}^{m\times n}$.

Theorem A.1: Consider system (A.1) and assume that

(i) the matrix A is asymptotically stable;

(ii) $B(0,0) = 0$, $D(0,0) = 0$ and $B(x,z), D(x,z)$ are continuous in $(x,z) = (0,0)$.

Then

a) system (A.1) is stable; and

b) there exists $\epsilon, \gamma > 0$ such that if $||x(0)|| < \epsilon$ and $||z(0)|| < \gamma$, then $x(k)$ converges to zero as k tends to infinity.

Notice that, under the assumptions of Theorem A.1, the linearization of the nonlinear system (A.1) gives the linear system

$$\begin{bmatrix} x(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ z(k) \end{bmatrix}$$

which is only marginally stable, since it has some eigenvalues that are equal to 1. For this reason the theorem of linearization can not be applied. In order to prove Theorem A.1 we need a couple of lemmas.

Lemma A.2: Let A be an asymptotically stable matrix and let P be a positive definite matrix solution of the discrete Lyapunov equation $A^TPA - P = -I$. Then there exists a constant $\nu > 0$ (depending only on A and P) such that, if the matrix $B \in \mathbb{R}^{n\times n}$ satisfies the norm inequality $||B|| < \nu$, then

$$(A+B)^TP(A+B) - P \leq -\frac{1}{2}I \tag{A.2}$$

Proof: Observe that $(A+B)^TP(A+B) - P = B^TPB + A^TPB + B^TPA - I$ and so (A.2) holds true if and only if $B^TPB + A^TPB + B^TPA \leq 1/2I$. This happens if and only if, for all $x \in \mathbb{R}^n$, we have that

$$x^T(B^TPB + A^TPB + B^TPA)x \leq \frac{1}{2}||x||^2$$

Observe finally that

$$x^T(B^TPB + A^TPB + B^TPA)x \leq ||P|| \left(2\,||A||\,||B|| + ||B||^2\right) ||x||^2$$

and hence the assertion of the lemma holds true if we choose $||B||$ small enough that

$$2\,||A||\,||B|| + ||B||^2 \leq \frac{1}{2||P||}$$

■

In order to present the next lemma we first need to introduce the following time-varying discrete time linear system

$$\left\{ \begin{array}{rcl} x(k+1) & = & Ax(k) + B(k)x(k) \\ z(k+1) & = & z(k) + D(k)x(k) \end{array} \right. \tag{A.3}$$

where $x(k) \in \mathbb{R}^n$, $z(k) \in \mathbb{R}^m$, A, $B(k) \in \mathbb{R}^{n\times n}$ and $D(k) \in \mathbb{R}^{m\times n}$.

Lemma A.3: Consider the system (A.3). If the matrix A is asymptotically stable, then there exists a constant $\nu > 0$ (depending only on A) such that, if $||B(k)|| < \nu$ for $k = 0, 1, \ldots, \bar{k}$, then

$$||z(k)|| \leq ||z(0)|| + C \left\{ \max_{i=0,1,\ldots,\bar{k}} ||D(k)|| \right\} ||x(0)||, \qquad \forall\, k = 0, 1, \ldots, \bar{k}, \bar{k}+1 \tag{A.4}$$

where C is a positive constant depending only on A.

Proof: Let $P > 0$ such that $A^TPA - P = -I$. Then, for the previous lemma, there exists a constant $\nu > 0$ such that, if $||B(k)|| \leq \nu$ for $k = 0, 1, \ldots, \bar{k}$, then

$$(A + B(k))^TP(A + B(k)) - P \leq -1/2I, \qquad \forall\, k = 0, 1, \ldots, \bar{k}$$

On the other hand, there exists $\theta > 0$ such that $\theta P \leq 1/2I$ and so, letting $V(x) := x^TPx$ we obtain that

$$V(x(k+1)) = x(k)^T(A + B(k))^TP(A + B(k))x(k) \leq (1-\mu)x(k)^TPx(k) = (1-\mu)V(x(k))$$

for all $k = 0, 1, \ldots, \bar{k}$. From this we argue that

$$V(x(k)) \leq (1-\mu)^k V(x(0)), \qquad \forall k = 0, 1, \ldots, \bar{k}, \bar{k}+1.$$

Notice now that there exist $\theta', \theta'' > 0$ such that $\theta' V(x) \leq ||x||^2 \leq \theta'' V(x)$ and so

$$||x(k)||^2 \leq \theta'' V(x(k)) \leq \theta''(1-\mu)^k V(x(0)) \leq \theta''/\theta'(1-\mu)^k ||x(0)||^2$$

for all $k = 0, 1, \ldots, \bar{k}, \bar{k}+1$. Observe finally that

$$z(k) = z(0) + \sum_{i=0}^{k-1} D(i)x(i)$$

and so, for all $k = 0, 1, \ldots, \bar{k}, \bar{k}+1$, we have that

$$||z(k)|| \leq ||z(0)|| + \sum_{i=0}^{k-1} ||D(i)||\ ||x(i)|| \leq ||z(0)|| + \left\{ \max_{i=0,1,\ldots,\bar{k}} ||D(i)|| \right\} (\theta''/\theta')^{1/2} \sum_{i=0}^{\infty} (1-\mu)^{i/2} ||x(0)||$$

■

Proof of Theorem A.1 We start by proving the stability. Let P the positive definite matrix such that $A^T P A - P = -I$ and let $V(x) := x^T P x$. Fix any arbitrary positive constants ϵ, γ such that

- $B(x,z)$ and $D(x,z)$ are continuous in $\{(x,z) : ||x|| < \epsilon, ||z|| < \gamma\}$;
- $||B(x,z)|| < \nu$ in $\{(x,z) : ||x|| < \epsilon, ||z|| < \gamma\}$ where ν is the constant introduced in Lemma A.2.

We want to show that there exist positive constants ϵ'', γ'' such that, if $||x(0)|| < \epsilon''$ and $||z(0)|| < \gamma''$, then $||x(k)|| < \epsilon$ and $||z(k)|| < \gamma$ for all $k = 0, 1, \ldots$.

First let $\epsilon' < \epsilon$ and $\gamma' < \gamma$ positive constants such that, if $||x|| < \epsilon'$ and $||z|| < \gamma'$, then $||Ax + B(x,z)x|| < \epsilon$ and $||z + D(x,z)x|| < \gamma$. Let now $m := \min\{V(x) \ : x \text{ such that } \epsilon' \leq ||x|| \leq \epsilon\}$ and finally introduce positive constants $\epsilon'' < \epsilon'$ and $\gamma'' < \gamma'$ such that

- $V(x) < m$ for all x such that $||x|| < \epsilon''$;
- $\gamma'' + CM\epsilon'' < \gamma'$, where $M := \max\{||D(x,z)|| \ : \ x \text{ and } z \text{ such that } ||x|| \leq \epsilon' \text{ and } ||z|| < \gamma'\}$ and C is the constant introduced in Lemma A.3.

We prove now the following fact

FACT If $||x(0)|| < \epsilon''$ and $||z(0)|| < \gamma''$, then we have that

$$||x(k)|| < \epsilon' \text{ and } ||z(k)|| < \gamma' \text{ and } V(x(k)) < m \tag{A.5}$$

holds for all $k = 0, 1, \ldots$.

Proof: We prove by induction on k. It is clear that (A.5) holds for $k = 0$. Assume now that (A.5) holds for all $k = 0, 1, \ldots, \bar{k}$. We want to show that (A.5) also for $\bar{k} + 1$. Indeed, notice that

$$||x(k)|| < \epsilon, \quad ||z(k)|| < \gamma, \quad \forall k = 0, 1, \ldots, , \bar{k}$$

and so

$$||B(x(k), z(k))|| < \nu, \quad \forall k = 0, 1, \ldots, , \bar{k}$$

and hence, applying Lemma A.3, we can argue that

$$||z(\bar{k}+1)|| \leq ||z(0)|| + C \left\{ \max_{i=0,1,\ldots,\bar{k}} ||D(x(i), z(i))|| \right\} ||x(0)|| \leq \gamma'' + CM\epsilon'' \leq \gamma'$$

Moreover, since $||x(\bar{k})|| < \epsilon'$ and $||z(\bar{k})|| < \gamma'$, then $||x(\bar{k}+1)|| < \epsilon$ and, by applying Lemma A.2,

$$V(x(\bar{k}+1)) < V(x(\bar{k})) < m$$

Therefore $||x(\bar{k}+1)|| < \epsilon$ and $V(x(\bar{k}+1)) < m$ implies that $||x(\bar{k}+1)|| < \epsilon'$ and so the fact is proved. ■

This fact proves the stability. It remains to be proved that $x(k)$ converges to zero. Indeed, since $||x(k)|| < \epsilon$ and $||z(k)|| < \gamma$, then $||B(x(k), z(k))|| < \nu$ for all $k = 0, 1, \ldots$, and so, using the same arguments used in Lemma A.3, we can argue that $||x(k)||$ converges to zero exponentially.

REFERENCES

[1] S. Oh, L. Schenato, and S. Sastry, "A hierarchical multiple-target tracking algorithm for sensor networks," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 2197–2202.

[2] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communication of the ACM*, vol. 46, pp. 34–40, 2004.

[3] B. Hohlt, L. Doverty, and E. Brewer, "Flexible power scheduling for sensor networks," in *In IEEE and ACM International Symposium on Information Processing in Sensor Networks (IPSN'04)*, Apr. 2004.

[4] M. Amin and P. F. Schewe, "Preventing blackouts," *Scientific American*, pp. 60–67, 2007.

[5] N. M. Fresis, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 1352–1364, 2011.

[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] 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.

[8] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *IEEE Network*, vol. 18, pp. 45–50, 2004.

[9] B. Sundararaman, U. Buyand, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[10] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Processing Magazine*, vol. 25, pp. 81–97, 2008.

[11] Y. R. Faizulkhakov, "Time synchronization methods for wireless sensor networks: A survey," *Programming and Computing Software*, vol. 33, no. 4, pp. 214–226, 2007.

[12] 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.

[13] J. V. Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proceedings of the International Workshop on Wireless Sensor Networks and Applications (WSNA)*, San Diego, California, USA, 2003.

[14] S. Yoon, C. Veerarittiphan, and M. L. Sichitiu, "Tiny-sync: tight time synchronization for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 2, pp. 1–34, 2007.

[15] K. L. Noh, E. Serpedin, and K. Qaraqe, "A new approach for time sycnhronization in wireless sensor networks: Pairwise broadcast synchronization," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3318–3322, 2008.

[16] 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.

[17] H. Dai and R. Han, "Tsynch: a lightweight bidirectional time synchronization service for wireless sensor networks," *ACM Mobile Computing and Communications Review*, vol. 8, no. 1, pp. 125–139, 2004.

[18] F. Fagnani and S. Zampieri, "Average consensus with packet drop communication," *Siam Journal on Control and Optimization*, vol. 48, pp. 102–133, 2009.

[19] S. Patterson, B. Bamieh, and A. E. Abbadi, "Convergence rates of distributed average consensus with stochastic link failures," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 880–893, 2010.

[20] 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.

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

[22] 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.

[23] L. Schenato and F. Fiorentin, "Average timesynch: a consensus-based protocol for clock synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, 2011.

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

[25] ——, "Optimal synchronization for networks of noisy double integrators," *IEEE Transactions on Automatic Control*, vol. 56, no. 5, pp. 1146–1152, 2011.

[26] R. Carli and S. Zampieri, "Networked clock synchronization based on second order linear consensus algorithms," in *49th IEEE Conference on Decision and Control (CDC'10)*, Atlanta, December 2010.

[27] R. Carli and E. Lovisari, "Robust synchronization of networks of heterogeneous double-integrators," in *IEEE Conference on Decision and Control*, Hawaii,USA, December 2012.

[28] Z.Zhong, P.P.Chen, and T. Hel, "On-demand time synchronization with predictable accuracy," in *Proceedings of IEEE Infocom)*, 2009.

[29] G. Russo and M. di Bernardo, "Contraction theory and master stability function: Linking two approaches to study synchronization of complex networks," *IEEE Transactions on Circuits and Systems II*, vol. 56, no. 2, pp. 177–181, 2009.

[30] J. Carr, *Applications of centre manifold theory*. Springer, 1981.