Fiftieth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 1 - 5, 2012

Pulse Coupled Discrete Oscillators Dynamics for Network Scheduling

Saman Ashkiani, and Anna Scaglione
Department of Electrical and Computer Engineering
University of California, Davis
Email: {sashkiani, ascaglione}@ucdavis.edu

***Abstract*—The dynamics of coupled oscillators, first introduced in mathematical biology, have increasingly become the inspiration for solving wireless scheduling problems. The appeal lies in the fact that coupled oscillators models suggest remarkably simple and scalable policies to enforce temporal events patterns in the absence of central control. Most authors have studied the emergent network behavior of "desynchronization", i.e. the state in which nodes equally partition a time frame into individual slots. However, less has been said about using these dynamics for the assignment of discrete resources, and the outcome of discrete oscillators dynamics. This problem is important because transmission events in general cannot have arbitrary duration, due to modulation constraints. Our problem has many features in common with "quantized consensus" problems which will be highlighted in this paper.**

In particular, in this paper we provide a model for analyzing Pulse Coupled Discrete Oscillators (PCDO) dynamics. As we will describe, the PCDO can naturally nest beneath the Pulse Coupled Oscillators (PCO) synchronization protocol, to attain a common shared slotted time. The PCDO dynamics assign a set of consecutive PCO slots, in an arbitrarily long frame of PCO slots. Our analysis shows that the algorithm converges almost surely and provides a bound on the convergence time of the PCDO dynamics.

I. Introduction

The dynamics of Pulse Coupled Oscillators (PCO), first introduced in mathematical biology [1], [2], have increasingly become the inspiration for solving wireless synchronization [3], [4], [5], [6] and scheduling problems [7], [8]. They leverage two emergent network behaviors. One is that of PCO synchronization, which can be used in locally connected networks as a clock distribution mechanism. The other one is that of "desynchronization" is exploited to attain Time Division Multiple Access (TDMA) scheduling, i.e. the state in which nodes equally partition a time frame into individual slots [8]. Their appeal lies in the fact that coupled oscillators models suggest remarkably simple and scalable policies to enforce temporal events patterns in the absence of central control.

However, some of the underlying assumptions on PCO synchronization and desynchronization are not easy to reconcile with physical constraints and natural operational scenarios. The goal of this paper is to address some of these shortcomings, and combine PCO synchronization and desynchronization to attain two way synchronous communications in decentralized networks. We first propose and analyze a novel model for Pulse Coupled Discrete Oscillators (PCDO) dynamics. The PCDO is the result of nesting the Pulse Coupled Oscillators (PCO) synchronization protocol, providing global network clock to time the transmissions, underneath the PCO desychronization protocol, which is used to partition a frame containing several PCO slots among the network nodes. In fact, the PCDO dynamics result in the assignment of a set of consecutive PCO slots within a frame, which as closely as possible matches a TDMA allocation. Our analysis shows that the algorithm converges almost surely and provides a bound on the convergence time of the PCDO dynamics. We then discuss how PCDO can enable the half duplex transmission of feedback to attain collision avoidance in clustered networks. We show that the PCDO/CA allows the coexistence of multiple clusters, supporting conflict free TDMA schedules without central coordination.

A. Background and Motivation

Network synchronization and scheduling are important primitives in sensing and network control applications, but are often at odds with the simple asynchronous nature of Radio Frequency (RF) packet switched protocols and are rather cumbersome as a result. They can be classified into two categories: centralized and decentralized solutions. Centralized synchronization protocols rely on reference signals sent from an external clock distribution infrastructure on a different band (e.g., Global Positioning System (GPS)); a master node that has access to the clock information distributes the signal to a number of network slaves. This is the case, for example, of the Network Time Protocol (NTP) [9] and of many of its variants. Decentralized synchronization protocols, instead, generate network agreement on the time of an event. The reference event can be generated by an external infrastructure or by one of the network nodes (e.g., Reference Broadcast Synchronization (RBS) [10]) and synchronization is attained via average consensus on the time of such event [11].

Inspired by models in mathematical biology, PCO protocols rely on the detection time of a physical signal, called firing event, to advance a local timer whose expiration, in turn, triggers each node firing. In PCO synchronization only when the nodes fire at unison, the nodes stop updating and advancing their clocks, due to the fact that the channel is half duplex. In the computer science literature the proponents of PCO assume that the firing events correspond to packets that are sent through a random access protocol [8], with the

978-1-4673-4539-2/12/$31.00 ©2012 IEEE

PCO update working at the application layer. In the communications and circuit literature PCO events and transmission of PCO signals match. In [6] the authors propose to have overlay clock distribution network. Prior to this work, we proposed a cross-layer design where PCO signaling and data transmission are interlaced [12].

As mentioned before PCO primitives have been also used to solve scheduling problems and find a TDMA schedule in a wireless network, with all to all connectivity. TDMA methods can also be classified into centralized and decentralized algorithms. Typically, both methods require network synchronization. In the former class, a central unit assigns portions of a time frame to different nodes. In the latter class the network nodes reach an agreement on the time frame portions they have exclusive access to [13]. Centralized TDMA protocols typically distribute the clock information through the master node and they also require an extra control channel accessed with contention to perform the handshaking with the master. Decentralized graph coloring algorithms are, instead, the abstraction behind decentralized TDMA protocols (see e.g. [13] and [14]). Compared to traditional TDMA methods, PCO based algorithms for desynchronization, such as DESYNC [8], do not require network synchronization. The PCO timer lasts for the frame duration. In this case when the timer triggers a firing signal, this also marks the beginning of the node data transmission period. By sensing the firing of other nodes, each participant moves its timer so that the firing will be sufficiently away from them, until each node attains an equal portion of the frame. The idea has been extended to provide a Proportional Fair Scheduling (PFS) [7] to share the time domain based on each node's demand, rather than uniformly.

This paper provides a model to support synchronous two way communications using PCO signaling, filling the following gaps left by the prior art, namely: 1) How can PCO desynchronization divide the frame in the discrete time units that are necessary to enable a physical transmission? 2) How can PCO synchronization be interlaced with data transmission efficiently? 3) How can PCO desynchronization avoid hidden and exposed terminal problems?

Specifically, due to modulation constraints it is not possible to have signals with arbitrarily small durations, and hence it is not clear in practice how desynchronization could work modulo a finite time interval, which is what is needed to transmit. To address this problem, in this paper we provide a complete model combining what we call the Discrete Dithered Desynchronization (D^3SYNC) algorithm [15], to attain TDMA schedules over discrete resources, with a PCO synchronization scheme, to support the common slotted timing. We refer to this methods as Pulse Coupled Discrete Oscillator (PCDO). Note that PCDO provides a natural structure to interlace PCO synchronization and data transmission, thanks to the PCO scheduling, by solving the second problem mentioned above.

The other gap we fill is the management of interference in locally connected networks. We focus on the case of clustered networks and propose the PCDO with *Collision Avoidance* (PCDO/CA) algorithm, in which the PCDO time slots are used in half-duplex by cluster-heads, to echo back the firing signal from the PCO nodes in the cluster, thereby providing to the rest of the network the information that prevents hidden and exposed terminal problems.

The paper is organized as follows. In Section II the main properties of the PCO based synchronization and time scheduling methods are reviewed. PCDO time scheduling method is introduced as a discrete time scheduling method coupled with a PCO based synchronization method in Section III. In Section IV PCDO/CA protocol is introduced and the dynamics will be discussed for the special case with two clusters. Simulation results are shown in Section V and a conclusion is made on Section VI.

II. PCO BASED COORDINATION PROTOCOLS

In this section we review the main ideas behind the PCO methods for synchronization and desynchronization. For simplicity of exposition, in both cases we assume a fully connected network $\mathcal{N}$ (i.e. all nodes can hear each other) composed of N nodes.

A. PCO Based Synchronization

In PCO synchronization each node has an internal timer, whose duration is the desired period for synchronizing the nodes activities. Every time the timer completes a cycle the PCO node can fire its signal. Initially, the nodes timers phases are distributed at random and the protocol objective is to change the timer phase in response to firing events that are sensed to align them, so that all clocks will eventually tick at unison. The situation is illustrated in Fig. 1(a), where the spheres around the circle mark the current state of the timer and the circle length is equivalent to the period, which without loss of generality is assumed to be 1. If $\Phi_i(t) \in [0,1]$ is the phase variable corresponding to the node i's internal timer, in isolation, the timer evolution can be expressed as:

$$\Phi_i(t) = (\frac{t}{T} + \varphi_i) \mod 1, \tag{1}$$

where it is assumed that all timers have an equal period T, and the aforementioned initial time instants can be distinguished by variables $\varphi_i \in [0,1]$ for all $i = 1, \ldots, N$. Without loss of generality we relabel nodes in a descending order in their initial phases such that $0 \leq \varphi_N \leq \varphi_{N-1} \leq \cdots \leq \varphi_2 \leq \varphi_1 < 1$. The *firing signal* is sent whenever each node completes its period (i.e. node i fires at time $t = t'$ if $\Phi_i(t') = 1$). Based on (1), the phase variable is reset to zero right after the firing (i.e. $\Phi_i(t'^+) = 0$) and the timer starts over. The reception of the firing signal triggers an update by all recipients that are not firing at the same time[1]. If the timer expiration and the actual transmission time match, the useful information is embedded in the actual timing of the reception, rather than being encoded in the signal itself. Different local update rules lead to synchronization [2].

[1]In practice there is a delay between transmission and reception. To account for that the nodes are firing simultaneously if their transmission is within a window that is called *refractory period*, which is a bound on such delay

A special case used in [16] is the following update rule:

$$\Phi_i(t_j^+) = \min\{(1+\alpha)\Phi_i(t_j), 1\}, \quad (2)$$

where $\alpha > 0$ and $\Phi_j(t_j) = 1$ (i.e. node j has fired). If $\Phi_i(t_j^+) = 1$, then the node is said to be *absorbed* by the last node that fired (i.e. node j), and in a fully connected network the two nodes continue to fire at unison.

B. PCO synchronization convergence time

Let $x_i(t) = \Phi_i(t) - \Phi_{i+1}(t) \pmod 1$ be the time gaps between consecutive phase variables at time t. If there are no absorptions, the evolution of time gaps $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ in the network $\mathcal{N}$ after every node has fired (a *round*) can be modeled as in [16] as the following linear affine mapping:

$$\mathbf{x}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{v}, \quad (3)$$

where $\mathbf{x}^{(k)}$ denotes the time gaps after k-th round of firing, $\mathbf{M} = (1+\alpha)^N\mathbf{I}_N$, and $\mathbf{v} = -\alpha((1+\alpha)^{N-1}, \ldots, (1+\alpha), 1)^T$. Based on definition $\mathbf{x}$ is such that $\|\mathbf{x}\|_1 = 1$, and $0 \leq x_i \leq 1$. In each round $\mathbf{M}$ linearly projects this point onto the subspace $(1+\alpha)^N\|\mathbf{x}\|_1$, and then $\mathbf{v}$ move the projected point back to the subspace $\|\mathbf{x}\|_1 = 1$ again. All of the eigenvalues of $\mathbf{M}$ in the update equation (3) are outside the unit circle, thus the system has a single unstable fixed point $\mathbf{x}^* = \mathbf{v}/(1-(1+\alpha)^N)$. Starting from any arbitrary initial point $\mathbf{x}^{(0)}$, as k increases $\mathbf{x}^{(k)}$ departs from the fixed point toward the boundaries, unless $\mathbf{x}^{(0)} = \mathbf{x}^*$ which occurs with probability zero. The PCO timers' evolution at every round of firing is captured by (3) until for some j, x_j crosses the boundaries ($x_j < 0$) which corresponds to an absorption (x_j are set to be zero afterward). The network then becomes equivalent to a new network with less nodes. This process continues until the complete synchronization, a state in which all the time gaps are equal to zero except for the one which is 1. The following lemma provide an expression for the maximum number of rounds till the first absorption event:

Lemma 1. *Assume that the PCO synchronization method is running over $\mathcal{N}$ with update equations in* (2)*. Let the initial time gap vector $\mathbf{x}^{(0)}$ be such that $\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2 \leq \delta$. At least one time gap is set to zero in less than $1 + \frac{\log(\sqrt{2}/\delta)}{N\log(1+\alpha)}$ rounds of updates.*

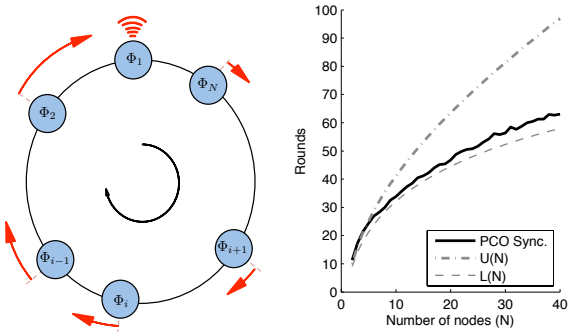
Proof: The proof for this lemma is in Appendix A ■

Assuming that the initial phases are chosen uniformly at random, it can be verified numerically (as it is shown in Fig. 1(b)) that $U(N) = \sum_{n=1}^{N}\mathbb{E}[1 + \frac{\log(\sqrt{2}/\delta)}{n\log(1+\alpha)}]$ is a good upper bound for the number of rounds needed until synchronization. It can be seen in Fig. 1(b) that $L(N) = U(N) - N$ is also a tight lower bound. Thus, the error of this estimation can at most be N in expectation.

C. PCO Time Scheduling or Desynchronization

Similar to the PCO synchronization, in PCO desynchronization each node has an internal timer which evolves in time as:

$$\Theta_i(t) = (\frac{t}{T'} + \theta_i) \mod 1, \quad (4)$$



(a) PCO updates. (b) Average number of rounds until the convergence of the PCO sync.

Fig. 1. Evolution of phase variables in the PCO sync. are shown in (a), and the convergence time of the PCO sync. averaged over 1000 number of independent trials are shown in (b). The approximation is made over computing the expected number of rounds needed over δ, and for uniformly distributed initial phase variables.

for $i \in \mathcal{N}$. Similarly we relabel nodes in a way that $0 \leq \theta_N \leq \theta_{N-1} \leq \cdots \leq \theta_2 \leq \theta_1 < 1$. In contrast to the synchronization, in this case rather than anticipating the timer towards the other firing events, their phases are updated so that their expiration is further away from the firing events of other nodes. In [8] the authors introduced the following update rule, called the DESYNC algorithm. Suppose node $i+1 \in \mathcal{N}$ fires at $t = t_{i+1}$, then node i will do the following update on its phase variable:

$$\Theta_i(t_{i+1}^+) = (1-\beta)\Theta_i(t_{i+1}) + \frac{\beta}{2}(\Theta_{i+1}(t_{i+1}) + \Theta_{i-1}(t_{i+1})), \quad (5)$$

where $\beta \in (0,1)$. Note that, strictly speaking, after the firing event of node $i+1$, node i knows that $\Theta_{i+1}(t_{i+1}) = 0$ but it does not have direct access to $\Theta_{i-1}(t_{i+1})$. In practice, node i will use as an approximation for $\Theta_{i-1}(t_{i+1})$ the last difference it recorded between the firing of node $i-1$ and its own (i.e. $\widehat{\Theta}_{i-1}(t_{i+1}) \approx 1 - \Theta_i(t_{i-1})$) . The objective of the update is to move node i clock closer by a factor β to the middle point between the two firing events before and after its own.

D. Desynchronization convergence time

Update equations of the DESYNC algorithm can also be represented in terms of the time gaps between two consecutive firings ($x_i \triangleq \Theta_i - \Theta_{i+1} \pmod 1$). Thus, the update equation in (5) can also be written in a matrix form as:

$$\mathbf{x}^{(k+1)} = (\mathbf{P}\mathbf{W})^N\mathbf{x}^{(k)}, \quad (6)$$

where $\mathbf{x}^{(k)} = (x_1, \ldots, x_N)^T$ is the time gap vector at the time right before the $(k+1)$-th firing of the node 1 (i.e. after k rounds). Also, $\mathbf{P}$ is a unit circular shift matrix, and $\mathbf{W}$ is defined as:

$$\mathbf{W} = \begin{pmatrix} 1-\frac{\beta}{2} & \frac{\beta}{2} & & \\ \frac{\beta}{2} & 1-\frac{\beta}{2} & & \\ & & \mathbf{I}_{N-2} & \end{pmatrix}. \quad (7)$$

This reveals the analogy between the DESYNC protocol and average consensus with pairwise updates [17].

It is shown in [8] that DESYNC converges asymptotically to the fixed point $\mathbf{x}^* = \frac{1}{N}\mathbf{1}$ (i.e. $\lim_{k\to\infty}\mathbf{x}^{(k)} = \mathbf{x}^*$). The algorithm converges to the ϵ-desynchrony neighborhood defined as $\|\mathbf{x}-\mathbf{x}^*\|_1 \leq \epsilon$ in $\mathcal{O}(N^2\log(1/\epsilon)/\beta)$ rounds (see [8] and [18]). In other words in $\mathcal{O}(N^3\log(1/\epsilon)/\beta)$ number of firings all nodes have their timers equally spaced around the unit circle.

III. PULSE COUPLED DISCRETE OSCILLATORS

In this section we introduce the PCDO algorithm. As we explain next, the PCDO is born from combining an underlying PCO synchronization algorithm, with period T_{PCO}, with a discrete PCO desynchronization process, working over a period that is an integer multiple of T_{PCO}. Like before, we assume that there are N nodes in the network $\mathcal{N}$ all hearing each others. Each node has a continuous PCO timer that evolves as $\Phi_i(t) = (\frac{t}{T_{\text{PCO}}} + \varphi_i) \mod 1$, for all $i \in \mathcal{N}$. Initially, the φ_i are not necessarily equal to each other, and hence nodes are not synchronized modulo T_{PCO}. These timers, named as *PCO timers*, set the pace for every other activity, which is done modulo T_{PCO}. There is also a discrete counter in each node, called the *PCDO counter*, and has a period equal to L slots, each of duration T_{PCO}. In isolation PCDO counters advances as:

$$\Upsilon_i[k] = (\frac{t}{T_{\text{PCO}}} + \sigma_i)|_{t=kT_{\text{PCO}}} \mod L, \tag{8}$$

where $\sigma_i \in \{0,1,\ldots,L-1\}$ is initially also random[2]. Equation (8) indicates that the PCDO counter increments one unit (i.e. $\Upsilon_i[k+1] = \Upsilon_i[k]+1$) whenever the PCO timer completes its cycle for the k-th time, i.e. at time $t = t_i^{(k)}$ such that $\Phi_i(t_i^{(k)}) = 1$. With respect to any arbitrary global time reference, the counters time evolution can be viewed as $\Psi_i(t) = \big(\Upsilon_i[k] + \Phi_i(t)\big)T_{\text{PCO}}$, for $kT_{\text{PCO}} \leq t < (k+1)T_{\text{PCO}}$. In the PCDO algorithm the end of the PCDO counter cycle is the event that triggers the emission of the firing signal to the network (i.e. the ith node firing occurs at step k such that $\Upsilon_i[k] = L$). In other words, in contrast with the original PCO method for synchronization, nodes do not transmit firing signals whenever their PCO timer ticks, they do it only when the PCDO counter overflows.

As before, the continuous PCO timers are updated exactly as in the standard PCO synchronization (2), every time a firing signal is sensed. Let $q_i[k] \triangleq \Upsilon_i[k] - \Upsilon_{i+1}[k] \pmod L$ be the time gaps between consecutive counters and let $\mathcal{Q}(x) = \min_{j\in\mathbb{Z}}|j-(x+\mathbf{v})|$ represents dithered quantization [19] for $x \in \mathbb{R}$ and $\mathbf{v} \sim \text{unif}(-\frac{1}{2},\frac{1}{2})$. Note that the quantity $q_i[k]$ is the number of time slots dedicated to node i for transmission at the times right after its overflow. Also $\sum_{i=1}^N q_i[k] = L$ and $q_i[k] \geq 1$ for all $i \in \mathcal{N}$ and for any $k \geq 0$. Then the update equation for the PCDO counter is a discrete and dithered version of the desyncrhonization update in (5):

$$\Upsilon_i^{\text{new}}[k] = q_i^{\text{new}}[k] = \mathcal{Q}\Big((1-\beta/2)q_i[k] + (\beta/2)q_{i-1}[k]\Big). \tag{9}$$

[2]If $L \gg N$, we can assume that initially no two nodes $i,j \in \mathcal{N}$ have equal PCDO counters

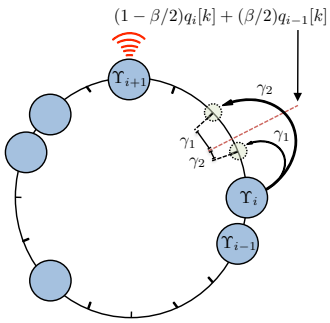


Fig. 2. The D^{3}SYNC updates (i.e. PCDO counter updates when all the PCO timers are synchronized). γ_1 and γ_2 represent the distance of the ideal update with the nearest integers (i.e. $\gamma_1+\gamma_2 = 1$)

In [15] we named the discrete algorithm performing updates according to (9) as *Discrete Dithered Desynchronization* (D^{3}SYNC) and its updates are illustrated in Fig. 2. The reason not to use the uniform quantization instead is to avoid getting stuck in any unfavorable fixed point.

The PDCO operates at two levels: 1) the PCO timers provide a synchronized time steps through the network; 2) the PCDO counters operate as a discrete version of the PCO desynchronization, the D^{3}SYNC, and subdivide the set of L slots as equally as possible among the nodes. The PCDO algorithm is summarized in Fig. 3.

PCDO scheme:

- If $\Upsilon_i[k-1] = L-1$ and at $t = \underline{t}$, $\Phi_i(\underline{t}) = 1$ for the k-th time then at $\underline{t}^+$:
  - Node i sends the firing signal to the network on its k-th time slot.
  - PCDO counter is reset $\Upsilon_i[k] = 0$.
- When the firing signal sent by node i is received at $t = \underline{t}$ then at $\underline{t}^+$:
  - All nodes $j \neq i$: $\Phi_j(\underline{t}^+) = \min\{(1+\alpha)\Phi_j(\underline{t}), 1\}$.
  - Node $i+1$ at its k-th time slot: $\widehat{q}_{i+1} = L - \Upsilon_{i+1}[k]$.
  - Node $i-1$ at its k-th time slot:
    - $q_{i-1} = \Upsilon_{i-1}[k]$.
    - $\Upsilon_{i-1}[k] = \mathcal{Q}((1-\beta/2)q_{i-1} + (\beta/2)\widehat{q}_{i-2})$.

Fig. 3. PCDO scheme which is done by all nodes.

Next we define more rigorously what we consider a valid Time Division Multiplexing (TDM) schedule when the time resources are discrete.

Definition 1. *Suppose there are N nodes in a network with L time steps in each period. Let $L = rN + \ell$, and $r \geq 0$, $0 \leq \ell < N$. All allocations with which ℓ nodes have a time gap equal to $r+1$ time slots, and others have time gap equal to r time slots are considered as a valid* TDM state.

A. Convergence of the PCDO algorithm

The PCDO convergence requires that both the PCO timers as well as the PCDO counters converge to a fixed point. The PCO timers are updated in a way that is identical to the PCO synchronization discussed in Section II-A. However the updates occur much less frequently, since the completion of a round now takes $L \gg N$ cycles, and that continues

to be true no matter how many nodes get *absorbed*. Note that in this case, synchronized PCO timers do not transmit at unison because their transmission epochs are marked by the PCDO counters, which tend to remain apart by as many PCO cycles as possible because of (9). Thus, the number of firing events that it takes to converge is exactly identical to that of the original PCO synchronization protocol and, therefore, the results stated in Section II-B are valid. This is useful, since a node joining the network can count a fixed number of their own firings prior to updating the PCDO, to ensure that their PCO timers are sufficiently close to being in synchronization. Hence, assuming all nodes count to a certain number of firings prior to updating their original PCDO counters (by waiting almost $L.U(N)$ based on the upper bound found in Section II-B), PCDO and D^{3}SYNC have analogous convergence properties.

Remark 1. The D^{3}SYNC is very similar to the so called *probabilistic quantized consensus* proposed and analyzed in [20]. The two protocols differ only in the way the updates are structured: for the dithered quantized consensus nodes wake up at random and communicate with random neighbors, while in D^{3}SYNC the interactions are cyclical and always with the same nodes.

Considering the properties of the D^{3}SYNC algorithm discussed in [15], it can be shown that the D^{3}SYNC algorithm converges to one of the TDM states almost surely and with $\mathcal{O}(\frac{2-\beta}{24\beta}N^3)$ expected number of interactions. The proof of almost sure convergence is very similar to that for *quantized consensus* in [21] while the proof of the convergence time is totally different and it is based on analyzing an associated Markov chain that describes the evolution of the network towards TDM states (see [15] for more details).

IV. Pulse Coupled Discrete Oscillators with Collision Avoidance

While PCO synchronization has been proven to converge in networks that are only locally connected ([4], [22]), all of the PCO based time scheduling algorithms described so far converge to a conflict free TDM schedule only in a fully connected network [8]. If the connectivity graph is not complete there can be collisions, caused by *hidden terminals*. Inefficiencies in transmissions can also arise due to *exposed terminal* problems. The former arises when two transmitters cannot sense each other, but are sensed by the destination of one or both of their transmissions, and they decided to transmit concurrently. The latter occurs when two transmitters sense each other but their destinations are not in the range of both, consequently their concurrent transmission is possible, but only one of them transmits. These are serious limitations in deploying these algorithms in real ad-hoc network scenarios.

In this section we leverage on the properties of the PCDO to introduce a Collision Avoidance (CA) feedback mechanism. We consider the case of *clustered networks* where nodes are grouped in clusters and where in each group nodes only want to exclusively transmit data to their own *cluster head*. Naturally, the assumption we make on the network connectivity is that each node in the cluster is heard directly by its cluster-head and vice versa.

Although it is assumed that clusters are independent from each other in their communications, nodes in different clusters may interfere with each other, as they try to communicate to their respective cluster heads. The goal of our protocol is to prevent that from happening, without requiring complex interactions among the cluster-heads. The *PCDO with Collision Avoidance* (PCDO/CA) protocol which we propose has these desirable features. The PCDO/CA protocol has two components: 1) a collision avoidance component, and 2) the time synchronization and scheduling component analogous to the PCDO. Like Carrier Sensing Multiple Access/CA (CSMA/CA) PCDO/CA relies on a feedback signal sent by the cluster-head that echoes back the firing signal from any node it hears (within or outside its own cluster). The method is similar in spirit to the handshaking mechanism in CSMA/CA that uses Request To Send (RTS) and Clear To Send (CTS) frames to avoid collisions and exposed terminals. Similarly, in PCDO/CA, the cluster head, by echoing back any firing signal it receives, informs all nodes about any possible interfering nodes. The basic difference compared to CSMA is that PCDO/CA is synchronous and the uplink firing and downlink echoes are sent in half-duplex during the same PCO timer period. Specifically, all nodes consider the PCO timer period divided in two parts: one is the uplink channel, dedicated to the transmission of the cluster nodes, the subsequent one is the downlink channel portion, dedicated to the transmission of cluster-head. Hence, the uplink firing signal and the downlink echoes follow each other at a synchronous pace. The time scheduling part of the PCDO/CA is done by performing the PCDO updates based on the information from the collision avoidance echoes from the cluster heads, rather than relying on firing signals from other nodes in the cluster. These aspects are discussed more in detail next.

A. Collision Avoidance component of PCDO/CA

Let assume that each node has an internal counter with period of L time steps each with a duration of T_{PCO}. At this point we assume that all PCO timers are already synchronized and all PCDO counters increment simultaneously (within the same T_{PCO}). As we said, each time slot duration T_{PCO} is divided into two portions: one for the uplink, from cluster nodes to cluster-heads, and one for the downlink, from cluster-heads to cluster nodes. The uplink is dedicated to nodes to send the firing beacons or data through the medium, and the downlink is dedicated to the cluster heads to send a feedback signal to all the hearing nodes. The access protocol works as follows:

- Each node broadcast a firing beacon (in the uplink channel) to the network whenever its PCDO counter overflows. After this point this node has the right to send data through the medium in the uplink if and only if it its own firing is followed by the downlink echo. If not, the node will have to start over, searching another white

space at random where to start again the PCDO protocol. In fact, in this case the cluster-head is experiencing a collision and therefore the time period is taken by some other node within another cluster. Note that if the node receives multiple signals in the downlink, it has to process them in the same way as if it was a valid acknowledgement from its own cluster-head. In fact, this is what prevents all clusters from interfering with each other.

- Upon reception of a single firing signal from any node, the cluster head sends back an echo in the downlink channel, to inform all the nodes that are in its range the channel is now yielded to a user. If the cluster head experiences a collision it does not respond.
- The time available for transmission to the node is the period that elapses between the reception of the downlink echo and the time of the next counter overflow. This time becomes predictable from the latest value for the downlink echo immediately following its own downlink echo.

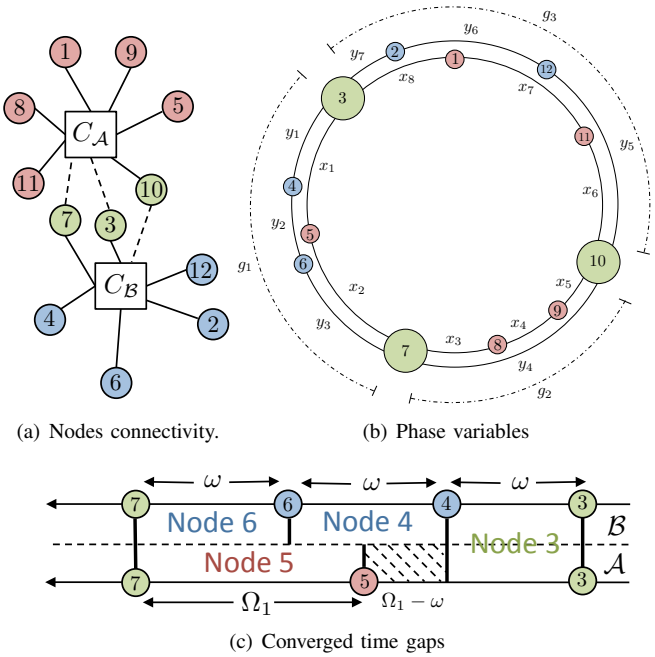
Next we specify how uplink firings and downlink feedback are used to advance the PCDO/CA algorithm.

B. Time Scheduling in PCDO/CA

The CA feedback implicitly informs all nodes about every possible neighbors of the cluster head, inside or outside its intended cluster. By utilizing an appropriate time scheduling scheme, which uses the mentioned proximity information about the network, a decentralized TDMA can be achieved as follows. The PCDO/CA algorithm performs a PCDO update as in (9) if and only if it receives a downlink echo. Otherwise, collisions are happening and therefore the node should start over from another white space available[3].

To understand how the network evolves towards a TDM schedule that allows clusters to coexist, in the following we discuss a scenario with two clusters. Specifically, suppose there are two clusters $\mathcal{A}$ and $\mathcal{B}$ with cluster heads $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ respectively. Let $|\mathcal{A}|$ denotes the number of nodes in cluster $\mathcal{A}$, and similarly $|\mathcal{B}|$ the number of nodes in cluster $\mathcal{B}$. We partition all nodes as follows $\mathcal{A} \cup \mathcal{B} = \{\hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{S}\}$, where $\hat{\mathcal{A}}$ (or $\hat{\mathcal{B}}$) are those nodes which can only hear $C_{\mathcal{A}}$ (or $C_{\mathcal{B}}$), and $\mathcal{S}$ denote all the *shared nodes* from cluster $\mathcal{A}$ (or $\mathcal{B}$), which are also in the range of $C_{\mathcal{B}}$ (or $C_{\mathcal{A}}$). There is not any difference between the shared nodes from the time scheduling point of view. In fact, regardless of their cluster association, they can be heard by both cluster heads and, hence, all the nodes in both $\mathcal{A}$ and $\mathcal{B}$ can cause a collision. Let us assume that L is chosen such that the TDM state (according to Definition 1) is unique. Next, we discuss how to determine the fixed point for any pair of clusters, like the ones shown in Fig. 4(a).

To gain some intuition on what are the possible fixed points, we now relax the assumption that the desynchronization is discrete, and look into the fixed points of the continuous time DESYNC explained in Section II-C which, however, is using the downlink echo signals (in this case ideally instantaneous) to make progress. The DESYNC algorithm tries to equalize the two time gaps before and after the time of firing signals. So, any pair of unequal time gaps that are in sequence, guarantees that the algorithm has not reach its steady state. Thus, the only possible fixed point would be the case where all pairs are equal. Similar to the DESYNC, in the PCDO/CA the nodes always update their phase variables so that it gets closer to the medium point between the phases of the two nodes that fire before and after (in this case signaled by the echo of the cluster head). Let us call these two nodes firing before and after the *time-neighbors* of a certain node. As long as the nodes hear the cluster head echo back a signal, their phase variables cannot cross each other. Thus, all nodes from $\hat{\mathcal{A}}$ or $\hat{\mathcal{B}}$ which are between two consecutive shared nodes in $\mathcal{S}$ will be trapped between them for all future updates. Therefore we call these periods *bins*, and examine more closely what happens within them. Specifically, each two shared nodes with consecutive firing times delimit a time bin g_k for $k \in \{1, 2, \ldots, |\mathcal{S}|\}$. Based on the nodes which fall in each bin, we can introduce the following partitions $\hat{\mathcal{A}} = \{\hat{\mathcal{A}}_1, \ldots, \hat{\mathcal{A}}_{|\mathcal{S}|}\}$ and $\hat{\mathcal{B}} = \{\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_{|\mathcal{S}|}\}$. We name *majority nodes* those forming the set that combines the shared nodes and the nodes that are the majority in each bin, i.e. $\mathcal{M} = \mathcal{S} \cup \bigcup_{k=1}^{|\mathcal{S}|} \arg\max_{\hat{\mathcal{A}}_k, \hat{\mathcal{B}}_k} \{|\hat{\mathcal{A}}_k|, |\hat{\mathcal{B}}_k|\}$. Similarly, we call *minority nodes* in bin g_k, the complementary set within the bin. The following lemma provides PCDO/CA.



(a) Nodes connectivity. (b) Phase variables

(c) Converged time gaps

Fig. 4. In this example $\mathcal{S} = \{3, 7, 10\}$, $\hat{\mathcal{A}} = \{1, 5, 8, 9, 11\}$ and $\hat{\mathcal{B}} = \{2, 4, 6, 12\}$. (a) shows the actual nodes connectivity in the network, (b) shows the phase variables, and (c) shows the dedicated time portions for each node after convergence. The wasted idle time is also shown.

Lemma 2. *In a network with two clusters, performing the PCDO/CA algorithm, there exists a fixed point such that the time gap between any two consecutive majority nodes is* ω,

[3]There are some complexities with this action that we leave out of this treatment for simplicity of analysis. However, these difficulties, that mostly arise due to the fact that the firing signals are only indicating starting times but not end times for the transmission, are solved by the PFS protocol proposed in [7].

and the minority nodes in bin g_k are spread over the bin with time gaps of Ω_k:

$$\omega = \frac{1}{|\mathcal{M}|}, \quad \Omega_k = \frac{\max\{|\hat{\mathcal{A}}_k|, |\hat{\mathcal{B}}_k|\}}{\min\{|\hat{\mathcal{A}}_k|, |\hat{\mathcal{B}}_k|\}}\omega, \tag{10}$$

where $|\mathcal{M}| = |S| + \sum_{k=1}^{|\mathcal{S}|} \max\{|\hat{\mathcal{A}}_k|, |\hat{\mathcal{B}}_k|\}$.

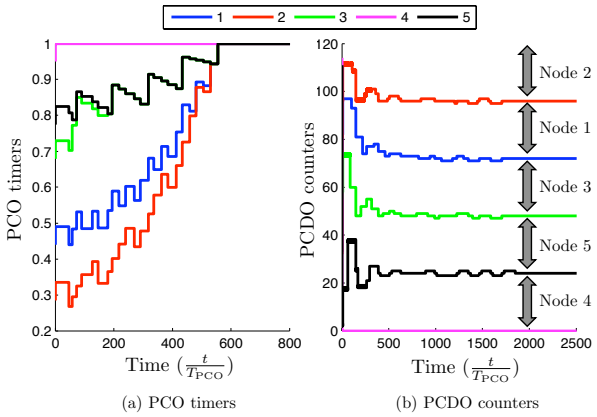
Proof: Assume that the time gaps are the ones stated in the lemma. We first consider the majority nodes $\mathcal{M}$. Those nodes which are not shared sense equal time gaps ω before and after their firing signal and, hence, they remain in their position, unchanged. For the shared nodes, although they may have two time neighbors from different clusters, as $\omega \leq \Omega_k$ for all k, then both of their time-neighbors are ω apart. Thus, shared nodes will not change their positions either. Minority nodes do not have any effect on the position of the shared nodes (because $\Omega_k \geq \omega$). Hence, as long as all minority nodes within a bin spread over the bin with an equal time gap Ω_k between each other, the state of the system remains unchanged. ■

In each update, two time gaps will change: the bigger one will shrink and the smaller one will expand. This trend will continue until they are, eventually, equal. Consequently, it is expected that after a while, if $|\hat{\mathcal{A}}_k| > |\hat{\mathcal{B}}_k|$ then node $\hat{\mathcal{A}}_k$'s phase will tend to be the closest to the boundaries of the period associated with the bin k, i.e the firing times of the shared nodes that delimit it. The reason is that they have to spread out more within the bin, given that they are more numerous. Our conjecture is that such a fixed point is the only fixed point of this system because of its similarity to the DESYNC.

As it is illustrated in Fig. 4(c) not all of the dedicated time gaps can be fully used by the nodes. The problem occurs when $|\hat{\mathcal{A}}_k| \neq |\hat{\mathcal{B}}_k|$ in an arbitrary bin g_k. In this case, the shared node which starts the bin, who starts sending data through the medium upon receiving the echo for its firing signal, transmits for a time $\omega < \Omega_k$; the minority node that follows will use the medium after $\Omega_k - \omega$, leaving an idle portion of time that is wasted, as it could be used by any of the nodes in the cluster of that minority nodes. Overall, the total wasted idle time once the network settles in the fixed point discussed in Lemma 2 is $T_{\text{wasted}} = \sum_{i=1}^{|\mathcal{S}|}(\Omega_k - \omega)$. It should be noted that the wasted time in each bin g_k will be minimized if $|\hat{\mathcal{A}}_k| = |\hat{\mathcal{B}}_k|$.

V. SIMULATION RESULTS

The PCDO algorithm converges to a state where all the PCO timers are equal within the network (i.e. synchronization) and all the PCDO counters are spread as much as possible to equally divide the entire period (i.e. L time slots) by reaching one of the TDM states (i.e. discrete desynchronization). In Fig. 5 the PCDO algorithm is used for a network with $N = 5$ nodes and $L = 120$ time slots. The initial values for all the timers and counters are chosen randomly. It should be noted that in this case a unique TDM state exists. The evolution of the phases of the PCO timers is shown in Fig. 5(a).



(a) PCO timers (b) PCDO counters

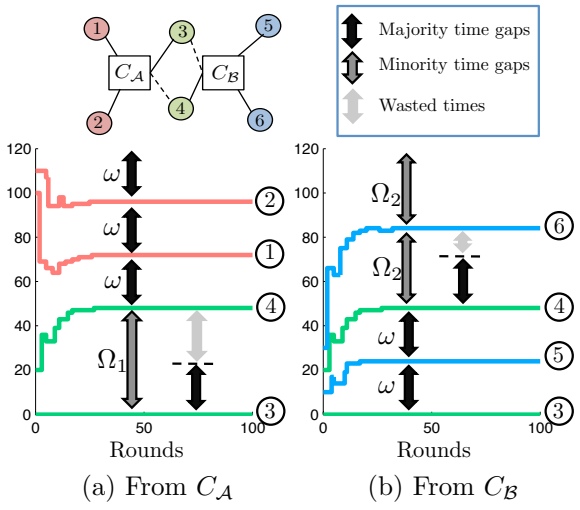
Fig. 5. The PCDO algorithm: The Evolution of the PCO timers and PCDO counters with time (values are depicted with respect to one of the nodes). The shown arrow are the dedicated time gaps to each node for data transmission. Other constants are as follows: $\alpha = 0.1$, and $\beta = 0.9$.

Note that in the implementation of the PCDO nodes compute the difference of their own counter and their time neighbors, by detecting their firing signals and looking at their local counters state as these events occur. Practically, this means that in updating their clock they directly access the phase of the node that fires before them, but they have to use an estimatation of the phase of the node that will fire after them, which is similar to what we indicated in (5). For example, when node $i-1$ is firing and node i receives it at its own k-th time stage, then if later node i receives node $i+1$'s firing signal at it k' time stage, $\widehat{q}_{i-1}[k'] = L - \Upsilon_i[k]$ is used instead.

The PCDO/CA protocol is simulated on a clustered network shown in Fig. 6. As it was mentioned in Section IV, the initial phases of the nodes (and especially their order with respect to the shared nodes) are effective in the final fixed point of the algorithm, and hence in the final time scheduling. In Fig. 6 the time schedule attained in the cluster from each cluster head's point of view (i.e. C_A and C_B) are shown respectively from left to right. The figure reports the time gaps between the majority nodes (referred to as majority time gaps), the time gaps between minority nodes (minority time gaps) as well as the wasted idle times. They match what Lemma 1 predicts. Based on extensive numerical trials we conjecture that, as long as the PCO clock is able to reach rapidly its stable fixed point, Lemma 1 is the only fixed point for the PCDO/CA algorithm with two clusters.

VI. CONCLUSION

In this paper we provided a completely decentralized algorithm for network synchronization and discrete time scheduling, which because of its properties it is more practical than the other existing desynchronization algorithms. The collision avoidance protocol was also proposed to encounter the hidden and exposed terminal problems in networks which are not all to all connected, and in particular are clustered. The fixed points of the PCDO/CA for a special case of two clusters was discussed, but a thorough analysis of the fixed points for the protocol is necessary. It is also favorable to distributively manage the number nodes from each cluster to



(a) From $C_\mathcal{A}$ (b) From $C_\mathcal{B}$

Fig. 6. Time scheduling by using the PCDO/CA algorithm with an initial counter values of $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_5 < \sigma_4 < \sigma_6$. Other constants are as follows: $L = 120$, $\beta = 0.9$. In this case $\omega = 24$, $\Omega_1 = 2\omega$ and $\Omega_2 = 3\omega/2$.

be as equal as possible between the shared nodes to minimize the wasted times.

Appendix A
Proof of Lemma 1

As $\alpha > 0$, all of the eigenvalues of $\mathbf{M}$ in (3) are $(1+\alpha)^N > 1$ and hence $\mathbf{M}$ is an unstable matrix. The system has a unique fixed point $\mathbf{x}^*$, but as long as the starting point is not exactly on the fixed point, $\mathbf{x}$ is getting further from $\mathbf{x}^*$ by each iteration. Also, by definition $\mathbf{x}$ is always on the subspace $\|\mathbf{x}\|_1 = 1$, and $0 \le x_i \le 1$. Thus, starting from any initial $\mathbf{x}^{(0)} \neq \mathbf{x}^*$, there exists a finite k' such that $\mathbf{x}^{(k')}$ falls outside the boundaries for the first time. Considering the inverse system in (3):

$$\mathbf{y}^{(k+1)} = \mathbf{M}^{-1}\mathbf{y}^{(k)} - \mathbf{M}^{-1}\mathbf{v}, \tag{11}$$

it is clear that $\mathbf{M}^{-1}$ is stable (i.e. $(1+\alpha)^{-N} < 1$) and it has the same fixed point $\mathbf{y}^* = \mathbf{x}^*$ as in (3). Thus, $\{\mathbf{y}^{(k)}\}_{k=0}^{\infty}$ is a sequence which converges asymptotically $\lim_{k\to\infty}\mathbf{y}^{(k)} = \mathbf{y}^*$.

$$\begin{aligned}\|\mathbf{y}^{(k)} - \mathbf{y}^*\| &= \|\mathbf{M}^{-1}\mathbf{y}^{(k-1)} - \mathbf{M}^{-1}\mathbf{v} - (\mathbf{M}^{-1}\mathbf{y}^* - \mathbf{M}^{-1}\mathbf{v})\| \\ &= (1+\alpha)^{-N}\|\mathbf{y}^{(k-1)} - \mathbf{y}^*\| \\ &= (1+\alpha)^{-Nk}\|\mathbf{y}^{(0)} - \mathbf{y}^*\|. \end{aligned} \tag{12}$$

$\mathbf{y}$ should satisfy the same conditions as $\mathbf{x}$, and hence $\|\mathbf{y}\|_1 = 1$ and $0 \le y_i \le 1$. However, the feasible domain for $\mathbf{y}$ forms a convex subspace and hence the euclidean distance of any two points within this convex subspace is less than the maximum euclidean distance of any two points on the boundaries, which in this case is $\sqrt{2}$. Thus, $\|\mathbf{y}^{(0)} - \mathbf{y}^*\| \le \sqrt{2}$ for any feasible $\mathbf{y}^{(0)}$ and hence (12) can be written as:

$$\|\mathbf{y}^{(k)} - \mathbf{y}^*\| \le \sqrt{2}(1+\alpha)^{-Nk}. \tag{13}$$

Thus, for any $\delta > 0$, $\|\mathbf{y}^{(k)} - \mathbf{y}^*\| < \delta$ for $k > \frac{\log(\sqrt{2}/\delta)}{N\log(1+\alpha)}$. Now, by considering again the main system in (3), $(k'-1)$-th round is the last round which falls inside the feasible region, and if $\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2 \le \delta$, then the inverse system in (11) reaches the δ-neighborhood of $\mathbf{y}^* = \mathbf{x}^*$ starting from $\mathbf{y}^{(0)} = \mathbf{x}^{(k'-1)}$ and after $1 + \frac{\log(\sqrt{2}/\delta)}{N\log(1+\alpha)}$ rounds.

References

[1] C. Peskin, "Mathematical aspects of heart physiology," *Institute of Mathematical Sciences*, 1975.

[2] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM J. Appl. Math.*, vol. 50, no. 6, pp. 1645–1662, 1990.

[3] Y.-W. P. Hong and A. Scaglione, "Time synchronization and reachback communications with pulse-coupled oscillators for uwb wireless ad hoc networks," in *IEEE Conference on Ultra Wideband Systems and Technologies (UWBST 2003)*, 2003.

[4] D. Lucarelli and I.-J. Wang, "Decentralized synchronization protocols with nearest neighbor communication," in *Sensys*, 2004.

[5] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05*, 2005.

[6] X. Wang and A. Apsel, "Pulse coupled oscillator synchronization for communications in uwb wireless transceivers," in *MWSCAS*, 2007.

[7] R. Pagliari, Y.-W. P. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE Journal on Selected Areas in Communications, Special Issue on Bio-Inspired Networking*, vol. 28, no. 4, 2010.

[8] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: Self-organizing desynchronization and tdma on wireless sensor networks," in *International Conference on Information Processing in Sensor Networks (IPSN)*, April 2007.

[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[10] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *SIGOPS Oper. Syst. Rev.*, 2002.

[11] Q. Li and D. Rus, "Global clock synchronization in sensor networks," in *INFOCOM*, 2004.

[12] Y.-W. Hong, A. Scaglione, and R. Pagliari, "Pulse coupled oscillators primitive for low complexity scheduling," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (Submitted)*, April 2009.

[13] T. Herman and S. Tixeuil, "A distributed tdma slot assignment algorithm for wireless sensor networks," *Algorithmic Aspects of Wireless Sensor Networks*, pp. 45–58, 2004.

[14] I. Rhee, A. Warrier, J. Min, and L. Xu, "Drand: distributed randomized tdma scheduling for wireless ad-hoc networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2006, pp. 190–201.

[15] S. Ashkiani and A. Scaglione, "Discrete dithered desynchronization," *(submitted to)* `arXiv [cs:NI]`, Oct. 2012.

[16] R. Pagliari, A. Scaglione, and Y. Hong, "Adaptive wireless networking primitives for distributed scheduling: Can radios swarm?" in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*. IEEE, 2009, pp. 133–136.

[17] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Conference on Information Processing in Sensor Networks*, Los Angeles, CA, April 2005.

[18] C.-M. Lien, S.-H. Chang, C.-S. Chang, and D.-S. Lee, "Anchored desynchronization," in *INFOCOM, 2012 Proceedings IEEE*, march 2012, pp. 2966 –2970.

[19] R. Wannamaker, S. Lipshitz, J. Vanderkooy, and J. Wright, "A theory of nonsubtractive dither," *Signal Processing, IEEE Transactions on*, vol. 48, no. 2, pp. 499–516, 2000.

[20] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, October 2008.

[21] A. Kashyap, T. Basar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[22] E. Mallada and K. Tang, "Synchronization of coupled oscillators," in *ITA*, 2010.