This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

PulseSS: A Pulse-Coupled Synchronization and Scheduling Protocol for Clustered Wireless Sensor Networks

Reinhard Gentz, Anna Scaglione, Lorenzo Ferrari, and Y.-W. Peter Hong

***Abstract*—The Pulse-coupled Synchronization and Scheduling (PulseSS) protocol is proposed in this work for simultaneous synchronization and scheduling of communication activities in clustered wireless sensor networks (WSNs), by emulating the emergent behavior of pulse-coupled oscillator (PCO) networks in mathematical biology. Different from existing works that address synchronization and scheduling (i.e., desynchronization) separately, PulseSS provides a coordination signaling mechanism that achieves decentralized network synchronization and time division multiple access scheduling simultaneously at different time scales for clustered WSNs. Here, we assume that the nodes are connected only locally via their respective cluster heads. Moreover, PulseSS addresses the issue of propagation delays, that may plague the accuracy of PCO synchronization in practice, by providing ways to estimate and precompensate for these values locally at the sensors (i.e., PCOs). At the same time the protocol retains the adaptivity and light-weight nature of PCO protocols both in terms of signaling and computations. Simulations of both the physical and the medium access control layers show a synchronization accuracy of factions of microseconds above 15dB of SINR for a 5 cluster network. A hardware implementation of PulseSS using TinyOS is also provided to corroborate the real world applicability of our protocol.**

***Index Terms*—Pulse-coupled oscillators, network synchronization, distributed time scheduling, clustered networks, medium access control, wireless sensor networks.**

I. Introduction

Communication in wireless sensor networks (WSNs) frequently rely on WiFi and Zigbee, which are inherently asynchronous and resolve medium access conflicts either through centralized management or through decentralized Carrier Sensing Multiple Access (CSMA) methods. The need for network synchronization, on the other hand, is typically addressed through out-of-band control channels, like the Global Positioning System (GPS), or through an application layer protocol such as the the Precision Time Protocol (PTP). While GPS and PTP can provide time information for synchronous sensing, they do not solve the communication scheduling problem which can be difficult, especially in large mesh networks of sensors. In fact, the optimal scheduling problem is known to be NP-hard [2] and thus, several heuristic solutions have been proposed to allocate portions of the time frame to network nodes, while meeting a given criterion of fairness [3] or maximizing data throughput. Protocols such as the USAP [4], DTSAP [5] or FLUSH [6] use a message-passing approach, while DRAND [7] and the method in [8] formulate the time scheduling problem as an instance of the graph-coloring problem. These scheduling algorithms typically rely on the availability of global synchronization and control information, such as the packet destinations and data-rates, to determine the conflict free schedule. However, in a large mesh network, sharing and computing this information may require significant overhead and complexity that increases rapidly with the network size.

For decades, engineers have tried to draw inspiration from the field of mathematical biology, to emulate the simple rules that lead to organized behavior in natural swarms and invent lightweight and easy to deploy protocols that meet the desired coordination goals. Our focus is a mechanism that builds on the *pulse coupled oscillator* (PCO) models from mathematical biology [9], and supplies two important primitives for clustered ad-hoc networks: decentralized synchronization and medium access control. The proposed protocol is thus referred to as the Pulse-coupled Synchronization and Scheduling (PulseSS) protocol. PulseSS works in an *ad-hoc* mesh network scenario, where nodes are grouped into clusters (with nodes communicating with a designated cluster head (CH)) and contend for the same spectrum resources adaptively. The architecture is similar to the IEEE 802.11 standard: 1) transmissions are only allowed from and to the CH and 2) CH's acknowledge the reception of signals from nodes in their range to expose hidden terminals.

A commonly used *centralized* protocol to similarly attain synchronization and scheduling with widespread acceptance for WSNs is WirelessHART [10]. Scheduling in WirelessHART is centrally managed by a single network manager, limiting the size of the application and introducing a single point of failure. In contrast, in our proposed protocol each cluster is managed locally, thus our solution is naturally scalable. Moreover, WirelessHART requires global knowledge of the network topology, whereas in our proposed protocol the nodes of each cluster will assign themselves a fair share by communicating locally within their cluster. The protocol ISA100.11a [11], is very similar to WirelessHART and has its key differences in the network layer and above. Therefore ISA100.11a suffers from the same problems as Wire-

R. Gentz, A. Scaglione, and L. Ferrari ({rgentz, anna.scaglione, lferrari}@asu.edu) are with the Department of Electrical Computer and Energy Engineering at Arizona State University, Tempe, USA. Y.-W. P. Hong (ywhong@ee.nthu.edu.tw) is with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan. Preliminary results of this work were presented at the IEEE Internet of Things World Forum 2015 [1]. This material is based upon work supported in part by the Department of Energy DE-OE0000780 and NSF CCF-1011811 (2010-2013).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

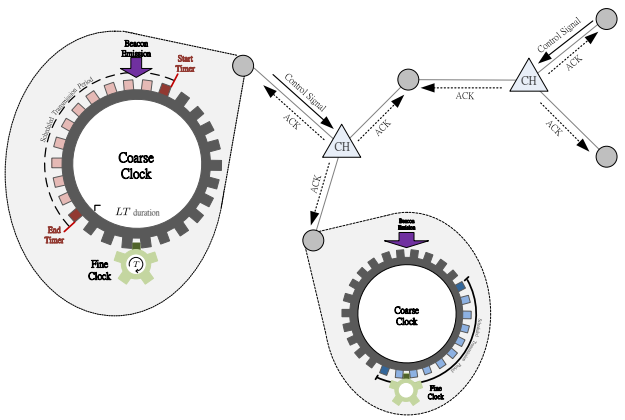

Fig. 1. A network with two clusters, with one node in range of both cluster heads and a pictorial representation of the coarse and fine clocks maintained by the nodes.

lessHART: central management and need for global knowledge of the network topology.

Related work on PCO-based scheduling algorithms include the DESYNC protocol in [12], our previous work on proportional fair scheduling [13], and follow-up works in [14], [15] that relax the all-to-all connectivity assumption required by the former two. PCO synchronization for networking applications was investigated by many, see e.g. [16]–[19]. Further work has shown that PCO synchronization does not work well if it is merged with CSMA/CSCA protocols [20]. Efficient implementations of the PCO protocol should disable CSMA [17] or use a separate dedicated radio band [21].

The key difference of this work is that PulseSS interlaces the PCO signaling with the scheduling signals, allowing to naturally separate the control traffic from the data traffic transmitted on the same physical channel. Compared to [12] and [13], the scheduling protocol we propose resolves conflicts among neighboring cells, i.e. it does not require an all-to-all connectivity. In fact, the PulseSS protocol can be viewed as the integration and realization of the theory of PCO synchronization and desynchronization previously studied in e.g. [12]–[19]. The benefits that we will showcase are: 1) collision avoidance; 2) integrated signaling; 3) improved synchronization accuracy. Some of our preliminary results on the PulseSS TinyOS experiment in the numerical section can be found in [1] and the related in depth convergence analysis is in [22].

The remainder of this paper is organized as follows. In Section II we give an overview of the protocol signaling and updates and the system model. In Sections III and IV, we describe the proposed PCO synchronization and scheduling protocols, respectively. In Section V we show the results of the protocol numerical simulations and in Section V-B we report the performance of the protocol in wireless network experiments that corroborate our hypotheses and findings.

II. OVERVIEW OF THE PULSESS PROTOCOL

Let the WSN be described by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of stationary sensor nodes and $\mathcal{E}$ (i.e., the set of edges) captures the pairs of nodes that are in range of each other. The network consists of a set of cluster heads (CHs) denoted by the set $\mathcal{C} \subset \mathcal{V}$ and a set of regular nodes $\mathcal{N} \triangleq \mathcal{V} - \mathcal{C}$ that communicates only with the CHs. For each $c \in \mathcal{C}$, we define $\mathcal{N}_c \subset \mathcal{N}$ as the set of regular (non-CH) nodes that lie within the transmission range of CH c; and, for each $v \in \mathcal{N}$, we define $\mathcal{C}_v \triangleq \{c \in \mathcal{C} : v \in \mathcal{N}_c\}$ as the set of CHs that are within the transmission range of v.

In this paper we assume that CHs are preassigned, such that each node has at least one CH in communication range. Nodes that have multiple CHs in communication range are referred to as *shared* (or *gateway*) *nodes*. The management of these nodes is crucial to ensure that all neighboring clusters can self-organize and attain conflict free schedules.

In PulseSS, each node maintains two local clocks, namely a *fine clock* with period T and a *coarse clock* with period LT, as illustrated in Fig. 1. Each cycle of the coarse clock is advanced by the expiration of L cycles of the fine clock and each cycle of the fine clock represents a transmission time slot of duration T. The PulseSS signaling is used to locally update the phases of both clocks, as will be explained mathematically next. These updates synchronize the phases of the fine clocks at all nodes at the slot level (see Section III) and set the phases of the coarse clocks apart so as to schedule for each node a portion of the L time slots available in the frame, enabling proportional fairness and spatial reuse (see Section IV). These goals are achieved by having each node transmit two control signals, a preamble which we call the *start beacon* and a postamble, called *end beacon*, meant to reach neighboring CHs. As in the 802.15.4 MAC these two signals delimit the period allotted for the two way data transmission between a node and is CH. However the beacons emissions are controlled by the regular nodes and not the CHs and there is no contention in this interval. The times of emission of these beacons governed by the local coarse clock expirations (every frame); the reception of such beacons by other nodes triggers adjustments of their own coarse clocks (and, thus, their schedules) as the CHs' corresponding acknowledgment is received. The notion of being *coupled* through an acknowledgment is new in PCO based protocols, and it is the key ingredient to attain collision avoidance.

Mathematically, let the state of the local fine clock at node $v \in \mathcal{V}$ be described by the phase variable

$$\Phi_v(t) = \frac{t}{T} + \phi_v \pmod 1, \tag{1}$$

where t is the absolute time and $\phi_v \in [0, 1)$ is the offset of the clock relative to the absolute time origin. The phase variable increases from 0 to 1 linearly in each period and marks the portion of time that has elapsed within each time slot. Moreover, to determine its transmission schedule, node v maintains not one but two ascending timers for the coarse clock, i.e., a *start timer* and an *end timer*, as depicted in Fig. 1. The state of the start and the end timers can be described by the phase variables

$$\begin{aligned} \Psi_v^{(s)}(t) &\triangleq \frac{t}{T} + \phi_v + \psi_v^{(s)} \pmod L \\ &= s_v(t) + \Phi_v(t) \end{aligned} \tag{2}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

$$\Psi_v^{(e)}(t) \triangleq \frac{t}{T} + \phi_v + \psi_v^{(e)} \pmod L$$
$$= e_v(t) + \Phi_v(t), \quad (3)$$

where $\psi_v^{(s)}$ and $\psi_v^{(e)}$ are integer offsets of the timers and $s_v(t) \triangleq \lfloor \Psi_v^{(s)}(t) \rfloor$ and $e_v(t) \triangleq \lfloor \Psi_v^{(e)}(t) \rfloor$ are the indices of the start and end time slots. The timers expire when their respective phase variables reach the value L and are reset to 0 afterwards. As mentioned before, the expiration of the start and end timers marks the first and last time slots that node v is scheduled to transmit (for a duration of $[\Psi_v^{(s)}(t) - \Psi_v^{(e)}(t) \pmod L] = [s_v(t) - e_v(t) \pmod L]$) and the transmission begins with the start beacon and ends with the end beacon. The two control signals inform the CH that a node v in range is transmitting for that time. The corresponding acknowledgments by the CHs, called *start* and *end acknowledgments*, warn other nodes in range that the channel towards the CH is busy. By having each node u, that hears the acknowledgements, update the discrete portions of its own start and end timers to avoid overlap (i.e., $s_u(t)$ and $e_u(t)$), nodes avoid conflicts (c.f. Section IV). At the same time, synchronization is achieved by using the estimated emission times of these beacons modulo T to update the fine-clock phase $\Phi_v(t)$ (c.f. Section III).

Note that, even though each node update is based only on the acknowledgment of its CHs' in range, the synchronization information will eventually propagate through the whole network via the updates and firings of shared nodes. It is important to remark that a node v updates its start and end timers based on the acknowledgements that occur right before and right after its start and end beacons, respectively. These acknowledgements may belong to different CHs. In fact, as time elapses, these acknowledgements will most likely come from CHs of the densest clusters. This will be made clearer in later sections.

PulseSS exhibits the following three main features:

- **Synchronization** – The network is synchronized at the slot level, i.e., $\Phi_u(t) = \Phi_v(t)$, for all $u, v \in \mathcal{V}$.
- **Collision Avoidance:** The transmission schedules of all nodes within the neighborhood of the same CH are disjoint, i.e., for any $c \in \mathcal{C}$ and $u, v \in \mathcal{N}_c$,

  $$\Psi_v^{(s)}(t) - \Psi_v^{(e)}(t) \leq \Psi_v^{(s)}(t) - \Psi_u^{(s)}(t),$$

  where the above operations are modulo L.
- **Proportional Fair Scheduling:** Transmission schedules of nodes that are in the same cluster, with no other conflicts and that are transmitting in succession of each other, are proportional to their demands.

Moreover, in each time slot, we divide the duration T into uplink and downlink transmission periods with durations $T_u = \lambda T$ and $T_d = (1-\lambda)T$, respectively, where $\lambda \in (0,1)$. This approach not only enables two-way communication between each node and its CH, but also avoids conflict between the uplink and downlink transmissions of two exposed nodes, i.e., two nodes that are in range of each other but are transmitting and receiving simultaneously with different CH partners. Note that in the setup we assume that there is no direct CH to CH communications. However, they could occur if CHs were to operate also as normal nodes for a neighboring cluster.

III. PulseSS Synchronization Updates

Next we describe the clock update procedures required to achieve synchronization among all fine clocks in the network. It is interesting to point out that the synchronization relies only on the beacons and acknowledgments utilized for scheduling and, thus, comes at *no additional signaling overhead*.

A. Effect of Delay on Conventional PCO Synchronization

The basic idea of PCO synchronization is to have each node emit a pulse (or beacon) whenever its local phase variable, say $\Phi_v(t)$, reaches 1 and have each node update its local phase variable whenever it receives a beacon from another node. In the early studies of PCO synchronization, the proof of synchrony among the local phase variables of all nodes often relied on three ideal assumptions: (i) all-to-all connectivity, i.e., all nodes are in range of each other; (ii) periodic pulsing, i.e., a pulse (or beacon) is emitted each time the phase of the local fine clock reaches 1; and (iii) instantaneous communication, i.e., a pulse (or beacon) transmitted by one node arrives at all other nodes instantaneously.

Let $t^+ \triangleq \lim_{\epsilon \to 0} t + \epsilon$ indicate the time immediately following time t. Specifically, suppose that a beacon was emitted by node u at time t, and that every other node $v \neq u$ who has overheard the beacon and is not firing at the same time, updates its local phase variable so that the value at time t^+ becomes

$$\Phi_v(t^+) = \min\{(1+\alpha)\Phi_v(t), 1\}, \quad (4)$$

where $\alpha \in (0,1)$. By doing so, the local phase variable is advanced by an amount proportional to its current value. Under the ideal assumptions mentioned above, it has been shown in [23] that the PCOs will eventually converge to synchrony with probability 1. That is, for almost all initial conditions, there exists t_0 such that $\Phi_v(t) = \Phi_u(t)$, for all nodes u, v and for all $t > t_0$. For locally connected networks, PCO convergence has only been proven by considering asymptotically small coupling (i.e. small α) (see e.g. [18], [19]). An extension of the presented protocol could include a different choice for the dynamics of PCO update in (4) to speed up the protocol convergence, as done e.g. in [24].

Note also that to allow for propagation delays, several works (e.g) introduced the use of the *refractory period*, i.e. a short period of time, following the emission of the beacon at each node, during which its phase variable cannot be updated[1]. The refractory period prevents a node from being affected by other nodes whose pulses were emitted simultaneously (i.e., synchronized) but arrive later due to propagation delay. For typical PCO-type protocols , the duration Δ_{ref} of the refractory period must be chosen to be larger than twice the propagation delay (to accommodate for the echoing effect that may occur when a node's own firing triggers the firing of another node) plus the likely range of the arrival time estimation error. In

[1]This reduction in sensing time in the PCO protocol is also applied in other works with the idea of energy savings [25], [26]. Our protocol naturally reduces the sensing time because the node can sense only once a frame and adjacent (in time) beacons described in Section IV. Because the periodicity of the beacons of interest is much slower than the PCO period T the receiver is already mostly off.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

mathematical terms, let r be the time of reception of a pulsing event that occurred at time t. Then, the modified update rule is given by

$$\Phi_v(r^+) = \begin{cases} \Phi_v(r), & \text{if } \Phi_v(r) \leq \Delta_{\text{ref}} \\ \min\{(1+\alpha)\Phi_v(r), 1\}, & \text{otherwise.} \end{cases} \quad (5)$$

The size of the refractory period, that is necessary for the protocol convergence, must be chosen such that $\Delta_{\text{ref}} > 2(r-t)$ for all neighborhoods in the network.

B. PulseSS Synchronization with Delay Compensation

In PulseSS, synchronization is achieved by utilizing the time of reception of the start and end beacons also used for scheduling, thus re-using the same signaling. The updates of local fine clocks are always between regular nodes (shared or not) and CHs that are in their range.

In [27] it was shown that in PCO synchronization the error accumulates over multiple hops and, thus, can be bigger than the refractory period Δ_{ref} required to observe convergence. PulseSS mitigates this effect by having the nodes estimate and compensate for the propagation delays of acknowledgments in their updates. This is possible because, in PulseSS, a pulse is emitted by each node only once every cycle of the coarse clock (which has duration LT), instead of once every fine cycle (which occurs with period T), as done in conventional PCO synchronization. This means the firing event of each node is isolated from other events in the same cluster and the propagation delay between the firing node and the CH can be estimated and the update can compensate for that. Therefore only propagation delay estimation errors (see next subsection) accumulate, but not the transmission delay itself.

Moreover it was shown in [22] that the PCO synchronization algorithm converges for any tree network to fixed points such that the maximum synchronization error can be bounded by the sum of the timing errors along the hops of the longest path starting from the (this result was predicted and shown numerically in [28]). Even though there is no proof of convergence valid for an arbitrary clustered network, numerical evidence shows that the network always converges as long as α is sufficiently small.

Specifically, let $\hat{\tau}_{v,c}$ be the estimated propagation delay between node v and CH c. Suppose that node v emits a start beacon at time $t_v^{(s)}$ and is received at CH c at time $r_v^{(s)}$. Upon receiving the start beacon from node v, CH c first computes an estimate of the emission time of the beacon as $\hat{t}_{c,v}^{(s)} \triangleq r_v^{(s)} - \hat{\tau}_{c,v}$ and then updates its local phase variable so that the phase at time $\hat{t}_{c,v}^{(s)+}$ (i.e., the time right after $\hat{t}_{c,v}^{(s)}$) is

$$\Phi_c(\hat{t}_{c,v}^{(s)+}) = \begin{cases} \Phi_c(\hat{t}_{c,v}^{(s)}), & \Phi_c(\hat{t}_{c,v}^{(s)}) \leq \Delta_{\text{ref}}, \\ \min\{(1+\alpha)\Phi_c(\hat{t}_{c,v}^{(s)}), 1\}, & \text{else.} \end{cases} \quad (6)$$

Afterwards, CH c sends an acknowledgment in the downlink (DL) transmission period of the next time slot, i.e., at time $\hat{t}_c^{(s)} + \lambda T$, where $t_c^{(s)} \triangleq \min\{t > r_v^{(s)+} : \Phi_c(t) = 1\}$ is the beginning of the next time slot according to CH c's local clock. This acknowledgment is received by all nodes in its range. Suppose that the acknowledgment is received at node $u \in \mathcal{N}_c$ at time $r_c^{(s)}$. Node u computes an estimate of $t_c^{(s)}$ as $\hat{t}_{c,u}^{(s)} = r_c^{(s)} - \lambda T - \hat{\tau}_{u,c}$ and updates its local phase variable so that the phase at time $\hat{t}_{c,u}^{(s)+}$ is given by

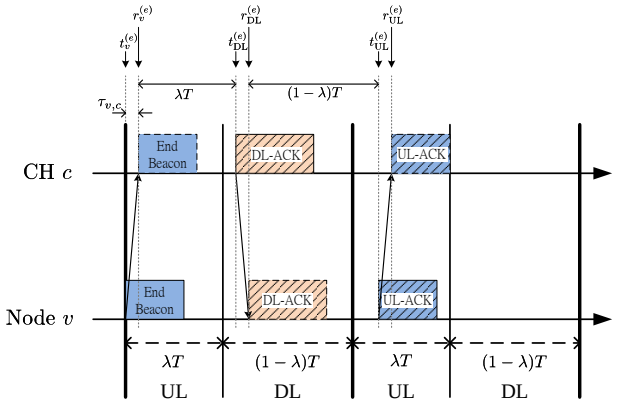

Fig. 2. Illustration of the signaling in the end time slot according to node v's time-scale used for delay estimation and scheduling.

$$\Phi_u(\hat{t}_{c,u}^{(s)+}) = \begin{cases} \Phi_u(\hat{t}_{c,u}^{(s)}), & \Phi_u(\hat{t}_{c,u}^{(s)}) \leq \Delta_{\text{ref}}, \\ \min\{(1+\alpha)\Phi_u(\hat{t}_{c,u}^{(s)}), 1\}, & \text{else.} \end{cases} \quad (7)$$

By observing the reception time of the acknowledgments, node u can also obtain an estimate of the emission time of node v's start beacon as $\hat{t}_{v,u}^{(s)} = \hat{t}_{c,u}^{(s)} - T$, which is used to determine the scheduling as described in Section IV.

C. Estimation of Propagation Delays

In the previous subsection we showed how the phase compensation in the synchronization updates (such as (7)) (which occur when nodes send their start beacons and the CH acknowledge them) can be performed with estimates of the propagation delays. In this subsection we describe how the estimation of the propagation delay $\hat{\tau}_{v,c}$ can be performed based on a specific hand-shaking protocol for the emission and acknowledgments of the end beacons. In particular we adopt a back-and-forth signaling between node v and CH c similar to that in PTP [29]. This is illustrated in Fig. 2. The impact on the network synchronization accuracy attainable with this technique is discussed further in Section V-A.

Let $t_v^{(e)}$ be the time that the end timer of node v expires and that an end beacon is emitted. Then the arrival time of the end beacon at CH c can be written as

$$r_v^{(e)} \triangleq t_v^{(e)} + \tau_{v,c} + z_v^{(e)}, \quad (8)$$

where $\tau_{v,c}$ is the propagation delay between node v and CH c and $z_v^{(e)}$ is the estimation error of the arrival time.

The estimate is carried out in the presence of noise and interference. In our simulations we model the estimation error $z_v^{(e)}$ as a Gaussian zero mean random variable with variance matching the so called Ziv-Zakai lower bound for the time of arrival error variance [30].

Upon receiving the end beacon, CH c waits for time λT and then emits, at time $t_{\text{DL}}^{(e)} \triangleq r_v^{(e)} + \lambda T$, an acknowledgment

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

on the downlink that arrives at node v at time $r_{\rm DL}^{(e)} \triangleq t_{\rm DL}^{(e)} + \tau_{v,c} + z_{\rm DL}^{(e)}$, where $z_{\rm DL}^{(e)}$ is the estimation error of the arrival time of the first acknowledgment at node v. Similarly, after time $(1-\lambda)T$ (i.e., at time $t_{\rm UL}^{(e)} \triangleq r_{\rm DL}^{(e)} + (1-\lambda)T$), node v emits a second acknowledgment on the uplink (UL) that arrives at CH c at time $r_{\rm UL}^{(e)} \triangleq t_{\rm UL}^{(e)} + \tau_{v,c} + z_{\rm UL}^{(e)}$, where $z_{\rm UL}^{(e)}$ is the arrival time estimation error made when processing the second acknowledgment at CH c[2].

Since $t_v^{(e)}$ and $r_{\rm DL}^{(e)}$ are known at node v, the propagation delay can be estimated at node v as

$$\hat{\tau}_{v,c}^{(v)} = \frac{1}{2}(r_{\rm DL}^{(e)} - t_v^{(e)} - \lambda T) = \tau_{v,c} + \frac{1}{2}(z_v^{(e)} + z_{\rm DL}^{(e)}). \quad (9)$$

Similarly, since $t_{\rm DL}^{(e)}$ and $r_{\rm UL}^{(e)}$ are known at CH c, the propagation delay can be estimated at CH c as

$$\hat{\tau}_{v,c}^{(c)} = \frac{1}{2}[r_{\rm UL}^{(e)} - t_{\rm DL}^{(e)} - (1-\lambda)T] = \tau_{v,c} + \frac{1}{2}(z_{\rm DL}^{(e)} + z_{\rm UL}^{(e)}). \quad (10)$$

For a static network, the delay estimation can be further refined by averaging over the estimates obtained in the recent M coarse cycles.

Note that, to ensure that the uplink and downlink transmissions are in their respective time periods, the durations of the end beacon, the DL acknowledgment, and the UL acknowledgment should be less than $\lambda T - \tau_{v,c}$, and $(1-\lambda)T - 2\tau_{v,c}$, $\lambda T - 3\tau_{v,c}$, respectively.

IV. PulseSS Scheduling Updates

Different from the synchronization updates, the scheduling of PulseSS for collision avoidance and proportional fair scheduling is achieved through the update of the discrete portions of the nodes' start and end timers (i.e., $s_v(t)$ and $e_v(t)$) in each cycle of the coarse clock. The updates and messaging mechanisms can be viewed as the realization of the theory of PCO desynchronization studied in [13] and are described as follows.

Specifically, suppose that the initial state of the start and end timers already satisfy the collision avoidance criterion, that is, for any $c \in \mathcal{C}$ and $u, v \in \mathcal{N}_c$,

$$\Psi_v^{(s)}(t) - \Psi_v^{(e)}(t) \le \Psi_v^{(s)}(t) - \Psi_u^{(s)}(t) \pmod L. \quad (11)$$

This can be achieved by letting the initial difference of the start and end timers at each node be sufficiently small. The expiration of the start and end timers marks the start and end of a node's transmission period in each cycle. Once the start (or the end) timer of a node, say node v, expires, a start (or an end) beacon is emitted by it in the UL period of the time slot. The beacon emitted by node v will then be acknowledged by all CHs in range, to inform all other nodes in the neighborhood of the CHs of the beacon emission. In case the collision avoidance criterion is violated, admission control at CHs would not acknowledge, i.e., may deny a second start beacon, before an end beacon is received, so that only one node at a time has channel access. We assume that all CHs in range acknowledge at unison with an identical beacon signal, such that acknowledgments are processed at the receiving node as a single signal affected by multi-path (we view this as a *cooperative-channel* acknowledgement).

Let $\mathrm{pre}(v) \in \cup_{c\in\mathcal{C}_v}\mathcal{N}_c$ and $\mathrm{suc}(v) \in \cup_{c\in\mathcal{C}_v}\mathcal{N}_c$ be the nodes that transmit immediately before and after node v, i.e., the *predecessor* and *successor* of node v. Node v adjusts its local timers in each cycle based on the expiration times of the end and start timers of nodes $\mathrm{pre}(v)$ and $\mathrm{suc}(v)$ respectively.

Let $t_v^{(s)} \in \{t : \Psi_v^{(s)}(t) = L\}$ be the expiration time instant of the start timer of node v in a given cycle of the coarse clock and let $t_v^{(e)} = \min\{t > t_v^{(s)} : \Psi_v^{(e)}(t) = L\}$ be that of the end timer of node v that follows immediately after. Moreover, let $t_{\mathrm{pre}(v)}^{(e)} = \max\{t < t_v^{(s)} : \Psi_{\mathrm{pre}(v)}^{(e)}(t) = L\}$ be the most recent expiration time instant of the predecessor's end timer and let $t_{\mathrm{suc}(v)}^{(s)} = \min\{t > t_v^{(e)} : \Psi_{\mathrm{suc}(v)}^{(s)}(t) = L\}$ be that of the successor's start timer. The corresponding time estimates[3] at node v are denoted by $\hat{t}_{\mathrm{pre}(v),v}^{(e)}$ and $\hat{t}_{\mathrm{suc}(v),v}^{(s)}$.

Immediately after receiving the acknowledgment to the start timer of $\mathrm{suc}(v)$, node v, at time $t_{\mathrm{suc}(v)}^{(s)+}$, updates its local timers in an attempt to move the discrete portion of the clocks phases (i.e., the time slot index) towards the target values

$$s_{v,\mathrm{target}} = \frac{D_v+\delta}{D_v+2\delta} e_{\mathrm{pre}(v)}(t_{\mathrm{suc}(v)}^{(s)+}) + \frac{\delta}{D_v+2\delta} s_{\mathrm{suc}(v)}(t_{\mathrm{suc}(v)}^{(s)+})$$
$$e_{v,\mathrm{target}} = \frac{\delta}{D_v+2\delta} e_{\mathrm{pre}(v)}(t_{\mathrm{suc}(v)}^{(s)+}) + \frac{D_v+\delta}{D_v+2\delta} s_{\mathrm{suc}(v)}(t_{\mathrm{suc}(v)}^{(s)+})$$

where D_v is a parameter capturing the demand of node v, δ is the portion of time slots reserved as guard period in between transmissions. If the target values are achieved, a portion of $D_v/(D_v+2\delta)$ of the time between the transmissions of its predecessor and successor is left for node v's transmission of its payload data and $\delta/(D_v+2\delta)$ portion of the time is left before and after its own transmission as guard intervals.

When $t_{\mathrm{suc}(v)}^{(s)}$ and $t_{\mathrm{pre}(v)}^{(e)}$ are perfectly known and that no updates have been made to predecessor's phase before time $t_{\mathrm{suc}(v),v}^{(s)}$, node v can infer that $s_{\mathrm{suc}(v)}(t_{\mathrm{suc}(v)}^{(s)+}) = 0$, since the timer must have reset to 0 after it has expired and that $e_{\mathrm{pre}(v)}(t_{\mathrm{suc}(v)}^{(s)+}) = (t_{\mathrm{suc}(v)}^{(s)+} - t_{\mathrm{pre}(v)}^{(e)})/T$, which is the time that has elapsed after the expiration of the end timer of node $\mathrm{pre}(v)$. However, in reality, these target values cannot be obtained precisely since only the estimates $\hat{t}_{\mathrm{suc}(v),v}^{(s)}$ and $\hat{t}_{\mathrm{pre}(v),v}^{(e)}$ are known at node v and also since the phase of the predecessor may in fact have been updated before time $\hat{t}_{\mathrm{suc}(v),v}^{(s)}$ due to the beacon emission of node v. In this case, node v can only obtain the estimated target values

$$\hat{s}_{v,\mathrm{target}} = \frac{D_v+\delta}{D_v+2\delta} \frac{\hat{t}_{\mathrm{suc}(v),v}^{(s)+} - \hat{t}_{\mathrm{pre}(v),v}^{(e)}}{T} \quad (12)$$

$$\hat{e}_{v,\mathrm{target}} = \frac{\delta}{D_v+2\delta} \frac{\hat{t}_{\mathrm{suc}(v),v}^{(s)+} - \hat{t}_{\mathrm{pre}(v),v}^{(e)}}{T}. \quad (13)$$

[2]Note that, in the case of a beacon sent by a shared node there are multiple CH that can respond. Our assumption is that the arrival time can be resolved as it is typical in multi-path channels, by estimating the strongest path arrival time which should correspond to the closest CH (if that is not the correct CH, a residual error will remain).

[3]Note that the time instants $t_{\mathrm{pre}(v)}^{(e)}$ and $t_{\mathrm{suc}(v)}^{(s)}$ can be estimated by node v through the reception time of CH's acknowledgments to these beacon signals, but the accuracy may be affected by synchronization errors and propagation delays, as described in the previous section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

Since the target values are not precise[4], it is necessary to further limit the adjustment of the timers at node v so that the relative order of its timers and the timers of its predecessor and successor are not altered, causing overlap in the schedules. This is achieved by further modifying their target values as

$$\tilde{s}_{v,\text{target}} = \min\left\{\hat{s}_{v,\text{target}}, \frac{s_v(\hat{t}^{(s)+}_{\text{suc}(v),v}) + \frac{\hat{t}^{(s)+}_{\text{suc}(v),v} - \hat{t}^{(e)}_{\text{pre}(v),v}}{T}}{2}\right\}$$

$$\tilde{e}_{v,\text{target}} = \max\left\{e_{v,\text{target}}, \frac{e_v(\hat{t}^{(s)+}_{\text{suc}(v),v})}{2}\right\}.$$

Finally, the local timers at node v are updated as

$$s_v(\hat{t}^{(s)+}_{\text{suc}(v),v}) = \mathcal{Q}\left[(1-\beta)s_v(\hat{t}^{(s)}_{v,\text{suc}(v)}) + \beta\tilde{s}_{v,\text{target}}\right] \quad (14)$$

$$e_v(\hat{t}^{(s)+}_{\text{suc}(v),v}) = \mathcal{Q}\left[(1-\beta)e_v(\hat{t}^{(s)}_{\text{suc}(v),v}) + \beta\tilde{e}_{v,\text{target}}\right] \quad (15)$$

where $\beta \in (0,1)$ and $\mathcal{Q}(\cdot)$ is a dithered quantization function [31] that maps the phase to the integer set $\{0,1,\ldots,L\}$ defined as $\mathcal{Q}(x) = \text{round}(x+v)$, where $v \sim \mathcal{U}(-1/2, 1/2)$. As shown in [32], the dithering operation ensures the convergence of the quantized consensus policy and has similar effects on PulseSS. In fact, as time elapses and synchronization is achieved, the dithered quantized desynchronization protocol mentioned above has been shown to converge for all-to-all networks in [33]. Its properties in a locally connected networks are discussed in the next subsection. To illustrate the idea an example on scheduling evolution is shown in Fig. 3.

A. Convergence properties of PulseSS

While the protocol always achieves a TDM schedule, the final solution may vary depending on the initial conditions. In some cases, certain portions of a frame corresponding to a particular cluster may be left empty in order to avoid conflict with neighboring clusters (an example in shown in Section V-B). We refer to these spaces left empty as *white spaces*. White spaces can be avoided under special conditions on the network topology and on the order of the initial clock phases, as specified in [22]. Note that, if the conditions on the topology are met, the network schedule has a unique fixed point but in general a different ordering will cause the schedule to loose efficiency. It is difficult to realize and enforce these conditions in general, nonetheless the conflict-free schedule provided by PulseSS generally supports higher network throughput compared to random access.

To explain the conditions for the existence of a unique efficient fixed point we define for each node i a unique cluster c (we will break ties if they exist by choosing an arbitrary one) with maximum demand as:

$$C(i) = \arg\max_{c' \in \mathcal{C}_i} \sum_{v \in \mathcal{N}_{c'}} (D_v + \delta). \quad (16)$$

Then, we define for each cluster c the sets of nodes:

$$\mathcal{A}_c = \{i \,|\, C(i) = c\} \quad (17)$$

[4]The time values used are based on possibly outdated information about the predecessor node's state at the time it last fired.

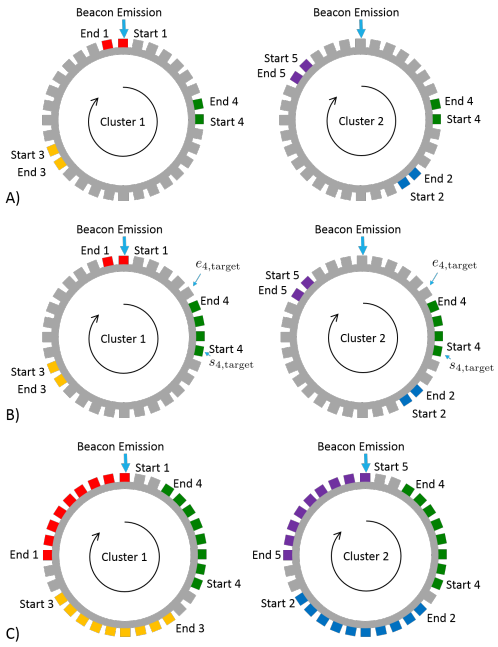

Fig. 3. Scheduling evolution illustration for a two cluster network with 4 local nodes{1,2,3,5} (2 in each cluster) and one shared node{4} as illustrated in Fig. 4. For this illustration, we overlay the scheduling of all nodes, assuming the nodes are synchronized and have equal demands. The shared node{4} is in both clusters, in green color, while the 4 local nodes{1,2,3,5}, 2 in each cluster, are each in their own color. Fig. A) shows the initial state of the system. We can see that each node only has the minimal size, in fact a slot for start and end beacon each. When time progresses and a start or end counter expires the node will send the corresponding beacon. Specifically, the start timer on the red node{1} is expired is Fig. B) and it sends out a beacon, that is acknowledged by the CH1. The green node{4}, notices this acknowledgment as the first start beacon following its own end beacon; Therefore finding that the red node{1} is its successor. Furthermore the green node{4} knows from previous firings that the blue node{2} is its predecessor. Then using (12)-(15) the green node{4} will calculate the target values for start and end timers and moves its start and end timers towards the target value, with a factor of β, claiming these transmission slots exclusively. Other nodes will not update their scheduling at this time as the red node{1} is not their respective successor. Then, as time continues, nodes will fire and update their timings based on the status of their respective successor and predecessor. In Fig. C) the nodes have reached a steady state, all have their proportional fair share, and most importantly the transmission of the shared node is respected in both clusters, so there is no channel access conflict.

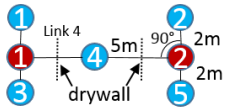

Fig. 4. Network topology of illustration in Fig. 3. The same topology is used for the simulation in Fig. 6, for which distances and drywall placements are marked.

$$\mathcal{S}_c = \mathcal{N}_c / \mathcal{A}_c \quad \forall\ c \in \mathcal{C}. \quad (18)$$

Since, by definition, for each node i its $C(i)$ is unique, the sets $\mathcal{A}_c$ form a partition of all the nodes in the network.

As specified in [22], a unique fixed point exists for PulseSS for any given *ordering* of the initial clock phases if the following condition holds:

1) For any c and $i, j \in \mathcal{S}_c$, it holds that $C(i) = C(j)$;

Furthermore, if such conditions are met then there exists an ordering that allows to have an efficient (i.e. white spaces free)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

Algorithm 1: Computation of the unique efficient fixed-point

% For a network with unique fixed point derives the
% unique schedule
find c such that $\mathcal{S}_c = \emptyset$
assign this schedule to all nodes v in $\mathcal{A}_c$:
$\mathcal{T}^c = LT$, $s_v(t) - e_v(t)$ in (19), δ_c in (20)
$\tilde{\mathcal{C}} = c$;
% $\tilde{\mathcal{C}} = \{c :$ nodes in $\mathcal{A}_c$ have an assigned schedule$\}$
while $\tilde{\mathcal{C}} \neq \mathcal{C}$ **do**
- pick a random $c \in \mathcal{C} \setminus \tilde{\mathcal{C}}$
- $[i_1, \ldots, i_{|\mathcal{S}_c|}] = \mathcal{S}_c$
- $c' = A^{-1}(i_1)$

if $c' \in \tilde{\mathcal{C}}$ **then**
assign this schedule to all nodes v in $\mathcal{A}_c$:
$\mathcal{T}^c$ in (21), $s_v(t) - e_v(t)$ in (22), δ_c in (23)
$\tilde{\mathcal{C}} = \tilde{\mathcal{C}} + c$;

schedule and Algorithm 1 calculates such efficient schedule. Other inefficient fix points can be calculated on a case by case basis. Interestingly there can be regions of convergence for the algorithm (i.e. non isolated fixed points) an example is offered in Section V-B) Further note that the condition above implies that it is possible to order the clusters in a tree structure where the *root* is the unique cluster c for which $\mathcal{S}_c = \emptyset$.

Let us now explain Algorithm 1. We assume that the conditions IV-A) and 1) are satisfied and denote by $\mathcal{T}^c \leq LT$ the portion of the frame available for the nodes in $\mathcal{A}_c$. For the unique cluster c, for which $\mathcal{S}_c = \emptyset$ we have $\mathcal{T}^c = LT$ and the portion of the frame available for all nodes $v \in \mathcal{A}_c$ is:

$$s_v(t) - e_v(t) = \frac{D_v}{|\mathcal{A}_c|\delta + \sum_{i \in \mathcal{A}_c} D_i} \mathcal{T}^c. \tag{19}$$

Furthermore each node in this unique cluster c has a guard space before and after its transmission time equal to

$$\delta_c = \frac{\delta}{|\mathcal{A}_c|\delta + \sum_{i \in \mathcal{A}_c} D_i} \mathcal{T}^c. \tag{20}$$

For all other clusters c', for which $\mathcal{S}_{c'} \neq \emptyset$, we will have:

$$\mathcal{T}^{c'} = \left[LT - (|\mathcal{S}_{c'}| - 1)\delta_{c'} - \sum_{v \in \mathcal{S}_{c'}} (s_v(t) - e_v(t)) \right] \tag{21}$$

and every node $v \in \mathcal{A}_{c'}$ will have

$$s_v(t) - e_v(t) = \frac{D_v}{(1 + |\mathcal{A}_{c'}|)\delta + \sum_{i \in \mathcal{A}_{c'}} D_i} \mathcal{T}^{c'}. \tag{22}$$

with a guard space before and after the transmission time:

$$\delta_{c'} = \frac{\delta}{(1 + |\mathcal{A}_{c'}|)\delta + \sum_{i \in \mathcal{A}_{c'}} D_i} \mathcal{T}^{c'}. \tag{23}$$

Algorithm 1 is a method to exhaustively calculate the schedule for all nodes by ensuring that $\delta_{c'}$ and $\sum_{v \in \mathcal{S}_{c'}} (s_v(t) - e_v(t))$ in (21) can be determined sequentially starting with the cluster with $\mathcal{S}_c = \emptyset$.

An example for $D_i = const.$ is shown in Fig. 5.

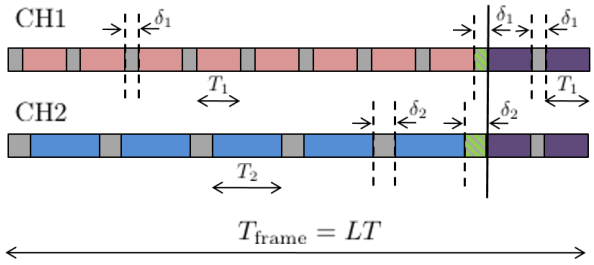

Fig. 5. Example of schedule in a two-cluster network using the topology with 2 shared nodes in purple and 7 and 5 local nodes respectively and all nodes with equal demand. We can see that cluster 1, has the highest total demand, since it contains more nodes, determines the size of the shared nodes. The other node's size are determined by each cluster individually. We can further find $T_1/\delta_1 = T_2/\delta_2 = D/\delta$ The guard band between local and shared nodes (in green) is determined only by the local nodes.

Notice that changes in demand are naturally leading to adaptively change the schedule. A similar effect is attained when nodes leave the network, because the total demand decreases and node can then occupy the empty space. All that remains to clarify is how new nodes join the network, which is the subject of the next subsection.

B. Node Joining and Admission Control

Suppose that node v wants to to join the network as a regular node. Before joining, node v first listens for at least one cycle LT to determine which CHs and nodes are within its neighborhood. We assume that node v is paired with specific CHs and if none of them is in range, node v will not join the network. If the intended CH is in range then node v checks after which node is the longest unoccupied space, that is large enough to fit a start and end beacon. in the next cycle node v listens for the end beacon (or, more specifically, the acknowledgment of the end beacon) of that node and emits its own start beacon $T + (1 - \lambda)T$ later (i.e., after the acknowledgments of the end beacon have died out according to Fig. 2). The end beacon is then send $2T$ later, resulting in no payload bytes in the first cycle. If the end beacon is acknowledged by the CH, then node v has successfully joined the network. On the other hand, node v will refrain from transmitting its end beacon if the start beacon is not acknowledged. Similarly, if more than one node is trying to join the network at the same time, the CH will not acknowledge the beacon emission due to a collision among the nodes that try to join, node v can try again at a another randomly chosen empty spaces.

V. Numerical Results

In this section, we demonstrate the performance of PulseSS in terms of both scheduling efficiency, convergence and synchronization accuracy.

A. Results on PulseSS Synchronization

In this section we evaluate by simulations the achievable synchronization accuracy of PulseSS. Here, we consider a network with 5 nodes and 2 CHs as illustrated in Fig. 4. The parameters used for this simulations are the given in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

TABLE I
SIMULATION PARAMETERS

Parameters	Value
Frequency	2.4GHz
Bandwidth	2Mhz
Tx & Rx Antenna height	1m
Temperature	300K
Rx Noise Factor	1
PCO-Period T	1/60s
Beacon length	6.4ms
Slots L	120
$(\alpha, D, \delta, \beta, \lambda, M)$	(0.04, 15, 7, 0.4, 0.5, 1)
Random Initialization	true

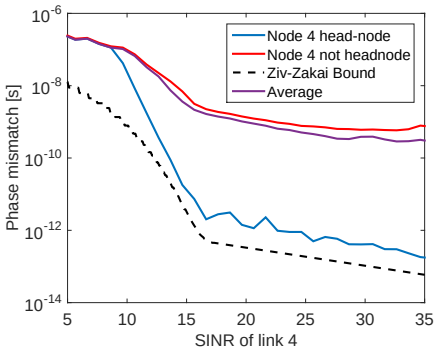

Fig. 6. Simulated fine clock accuracy, for node 4 (Fig. 4) being the leading (head node), or another node being the leading node. Note that link 4's SINR is equal to link 4's SNR as by protocol design it will exclusively transmit in it's timeslot.

We assume the receivers time of arrival estimation algorithm is reaching the accuracy predicted by the Ziv-Zakai Bound [30], which we evaluate as a function of the signal to interference and noise ratio (SINR). In fact, in spite of our conflict resolution technique, the reception will always be subject to some interference, which in short range communications will dominate over thermal noise. Thus, it is necessary to pay close attention to the interference and path loss attenuation. We calculate the interference power received from any active node to a certain receiver by scaling the transmit power of the active nodes as predicted by the indoor path-loss model for the ISM 2.4 GHz system proposed in [34]; the power level is also multiplied by a random exponential random variable with unit mean to model short-term fading. Using the signal to noise plus interference value of a certain link that is receiving a beacon signal, we simulate the time of arrival estimates at each receiver as the free space delay of the link plus zero mean Gaussian noise with variance equal to the Ziv-Zakai Bound evaluated at the link SINR level. The conflict graph is the topology shown in Fig.4. The shared node is placed between two drywalls, to illustrate that the clusters are typically at two separate locations providing an additional path loss attenuation of 5.7dB [34].

In Fig. 6 we plot the average absolute value of the phase mismatch (PM) of all nodes versus an arbitrary chosen reference as a function of the SINR of the link between node 4 and CH1[5], which is the minimum average SINR for the whole network. When the network converges all nodes in range are firing within the refractory period and there exist a certain order of firing. The PM shown in Fig. 6 is averaged among all nodes, over 1000 iterations, illustrates 3 cases: a) in the first case node 4 is the first in the final PCO firing order; b) in the second case one non-shared node is leading the others in the final firing order; c) in the third case the initial condition is chosen at random and either situation is possible, although not necessarily with the same frequency. In fact, it is clear that starting at random the final PCO firing order of case b) is quite frequent and the average performance tend to be closer to the worst ones. Another interesting observation (see also [28]) is that in PulseSS there is an accumulation of timing errors, adding up over the path length between the node firing first (the head-node) and the last, which explains the degradation compared to the Ziv-Zakai limit. What is most pronounced is the difference in accuracy we observe between the two cases mentioned earlier. In the first case both CH are absorbed by node 4, and can take advantage of the fact that PCO updates occur interference free by design. We can see that the accuracy in this case is close to the sum of the Ziv-Zakai limits of the links forming a 4 hop length path.

In the second case, the difference is that node 4 may update its clock in the presence of interference from one of the clusters. The results shown are averaged over all 4 possible choices for head-nodes other than node 4[6], which are identical due to symmetry. If two nodes from the two clusters that have node 4 in common transmit data while another is transmitting a beacon, assumed to have the same transmission power, then node 4 will experience interference from the concurrent transmission of data, resulting in the loss in accuracy in the time of arrival estimation that we observe in the simulations. Note, however, that a simple modification of the protocol would allow to attain the same accuracy as in case a all the time. In fact, instead of updating the PCO phase at every acknowledgement, nodes could selectively discard updates if the SINR are below a threshold (although coarse clock updates that require simply a correct detection would still have to be done). A more extreme option is that all nodes update exclusively when their specific acknowledgement are sent, once every frame. Of course, PCO updates would be far more infrequent and the convergence speed would be lower.

B. *Results on PulseSS Scheduling*

In this section we focus on the effectiveness of the PulseSS scheduling. The network topology we consider is in Fig. 7, bottom right with an average node to CH distance of 6.2m. To illustrate that nodes are in different rooms we place drywalls as in the previous section. The simulations parameters, Rayleigh fading and path-loss model are the same as in the previous section. Assume that the detection of a signal is perfect, but the time of arrival is affected by the same type of error as in the previous section. We are especially interested in the influence of noise and inaccuracies of the synchronization on the scheduling results, and also in the effectiveness and

[5]Both links from node 4 to the CH's are symetric and therefore identical.

[6]CH's never initialize a transmission and can never be head nodes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

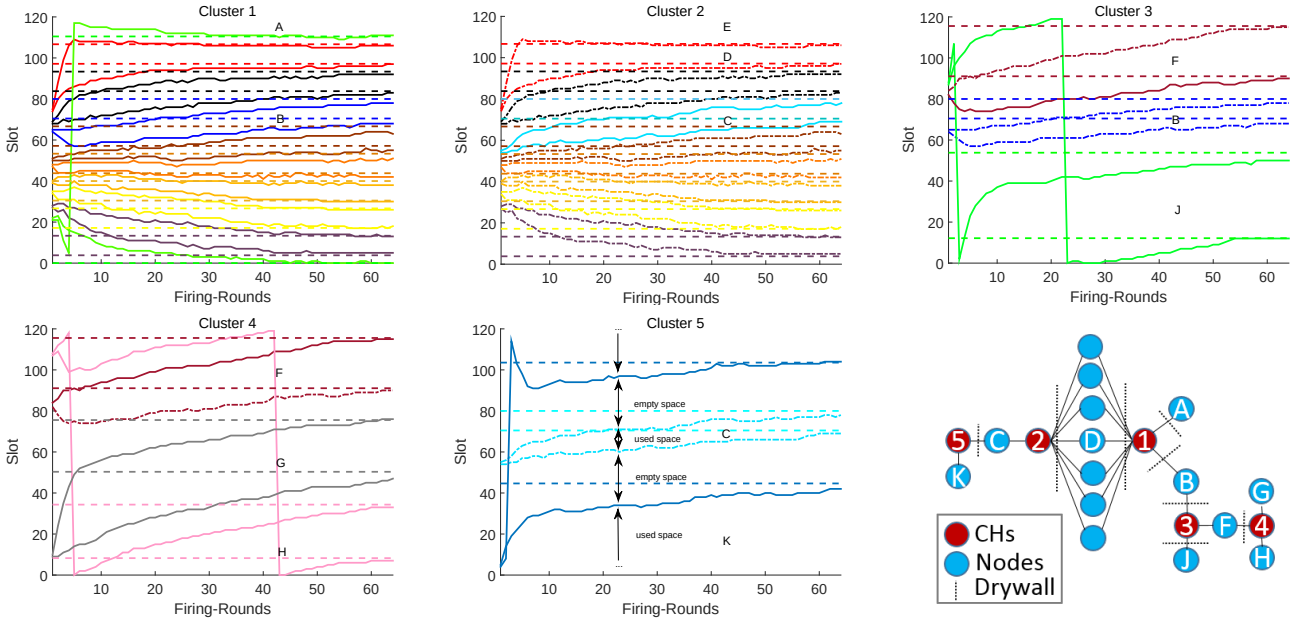

Fig. 7. Simulated TDMA Scheduling Result. The topology simulated and the location of nodes of special interest can be seen in the bottom right.

proportional fairness of PulseSS. In the simulations, all nodes are assumed to have equal demand.

In Fig. 7, we plot the state of the start and end timers of different nodes in a cluster with respect to the number of firing rounds. The values are shifted relative to the firing time of the start beacon of an arbitrarily chosen node in the cluster. The state of the start and end timers of the same node are plotted, in all clusters where the node is present, using the same color. The line style is solid, if the node updates with that cluster and dash-dotted if the nodes is not updating the respective timer with that cluster but is in range of the CH. The theoretical convergence points in Algorithm 1 are shown as the dashed lines for comparison. Note that the plot is shown modulo L, thus the slot $L-1$ and 0 are next to each other.

We can see, in Fig. 7, that the nodes have a small gap initially at their disposal, since they are initialized with the minimum possible distance between start and end clock and, as the time advances they acquire a fair amount of the frame. Notice that the sudden jump in phases of node A in cluster 1, node J in cluster 3, node H in cluster 4, and node K in cluster 5 are due to the artifact of the modulo L definition of the phases, i.e., due to the fact that slots L-1 and 0 are next to each other. We can clearly see that cluster 1 contains the most nodes and is therefore the densest cluster (*root*). As consequence, all nodes in range update with that cluster. Cluster 2, which is the second densest cluster, has only one node (node C) that is not shared with cluster 1. It can be seen that the schedule of cluster 2 is constrained by cluster 1 and the only node that is not in common, node C, reuses the same slot of node B from cluster 1. Note that there the presence of white space (marked by the letter E) in cluster 2, as we mentioned in Section IV-A. Equation (21) in Algorithm 1 is then modified to consider the effective $\mathcal{T}^2$ that node C has, i.e. an additional δ_1 and the schedule of node A are removed from $\mathcal{T}^2$. The unused space could have been filled if nodes A and B were next to each other in time, putting all shared nodes in consecutive spots (mod L) and requiring no adjustment of Algorithm 1. We can see that all nodes start and end timers approach their theoretical convergent points predicted by Algorithm 1 with the modification just described for cluster 2. The average channel utilization at the end over all clusters is 62%.

In cluster 5, we see one shared node C with cluster 2, that is, always updating with cluster 2 and therefore limited by cluster 2, visualized as a dash-dotted line. However, node K in cluster 5 has only one CH in range and, thus, can take advantage of the available space in that cluster and expand. We can further see that node K (seen in the outer part of the figure, due to the modulo L property), leaves a larger separation, potentially allowing node C to claim that space, since it is unaware of the limitations in another cluster.

In clusters 3 and 4, we can see the combination of all the previously mentioned effects. In cluster 3, node B is updating with cluster 1 and therefore not claiming any space. Node F's start beacon is confined by node H in cluster 4, but its end beacon is confined by node B in cluster 3. This is the best case scenario for the region of possible spaces between B and F that give a fixed point as discussed in [22], since condition 1 is violated for cluster $c=3$ and the mapping $C(i)$ in (16) that gives $C(B)=1$ and $C(F)=4$. The confinement due to node H is because in cluster 4 the nodes can share the available space equally. As a consequence the guard space between nodes in cluster 4 is smaller than those in cluster 3 where node B is confined by cluster 1. On the other end, node F's end beacon is confined by node B because that node is already fixed and cannot move. As a consequence all nodes from cluster 4 reduce their shares such that the limitations inflicted from node B, in cluster 3 are respected, and proportional fair scheduling is achieved. Node J behaves similarly to node K in cluster 5: it claims the available space but does not influence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

nodes B or F as they are confined by other clusters.

It is interesting to note that the convergence lines plotted in Fig. 7 help us visualize where the phases of the nodes start and end beacons will eventually converge as time goes to infinity. These phases need not stay above or below their respective convergence lines during the transience period. What is common to the evolution of all phases is that (i) the difference between the phases of the start and end beacons of each node will gradually expand towards occupying its proportional share of the frame, and (ii) once the expansion reaches a certain point, the phases of the beacons will start to shift in the same direction towards their convergence lines due to the pressure imposed by nodes that are firing just before or after them. This can be observed for all nodes. For example, for node K, the phases of its start and end beacons initiate at values in between their convergence lines (modulo L), but the difference between these phases rapidly expands (since node C is occupying only a small portion of the frame), causing the phase of its end beacon to go below its convergence line around the 4th firing round. These phases then both shift in the same direction towards their convergence lines due to the pressure imposed by node C, which is governed by the evolution of nodes in cluster 2. Similarly, for node J, the phases of its start and end beacons both initiate at values below their respective convergence lines (modulo L), and then shift in the same direction towards these lines.

VI. Microcontroller Implementation of PulseSS

We implemented PulseSS in a micro-controller equipped with a radio. In our implementation we can only access events reported by the radio to the micro-controller and this limits our precision in time to the time interval between clock ticks. The micro-controller is running with 7.3728MHz, which is not enough in an indoor environment to account for signal traveling delays, as the minimum step size is one clock cycle (=41m free space travel). This has prompted us to ignore the delay compensation feature for this implementation, which also eliminates the need for the third beacon acknowledgment. However with full implementation of the physical layer our compensation would be feasible. Furthermore, because we cannot modify and add physical layer signals and detection schemes, beacons need to be differentiated through a message in the payload that is decoded.

As we have a real time system we have have to account for CPU limitations and ensure that all calculations can be completed before the node or CH has to send the next command. The time that are needed to send beacons and acknowledgements, as shown in Fig. 8, including the computation times, need to fit in the PCO slot duration. To gain some extra computation time before the deadline arrives we can take advantage of the following: Because the CPU is notified when the radio chip receives a preamble, we can preemptively compute the updates, assuming the acknowelgement is valid. Once the complete packet is received (we ignore the stopbyte), either the type of beacon sent or the reception of data is confirmed, and the updates can be applied or discarded.

Like in TDMA each node can transmit once every LT. While the minimum size L depends on the maximum number of nodes in each cluster, we find the minimum period T is bounded below by the transmission delays and the devices computation speed. From the specification of our micro-controller we determined that the delay for a single transmission for our system is $t_{single} \approx 1099.176\mu$s, limiting the minimum round trip time. Therefore, based on the time required to transmit, receive and process the back and forth of beacons and acknowledgement (see Fig. 8) we find

$$T > \max(2t_{single}, t^{ch}_{compute} - t_{stop}, 0.5(t^{node}_{compute} - t_{stop}) \quad (24)$$

where $t_{compute}$ and $t^{ch}_{compute}$ are the nodes and CH's respective required computation times and t_{stop} is the transmission time of the stop-bit. We note that we have to include t_{single} in our calculation and treat it as a known transmission delay the same way as we do for $\hat{\tau}^{(s)}_{v,c}$ in (6) since we are interested in the time the beacon transmission was initiated rather than when it was completed.

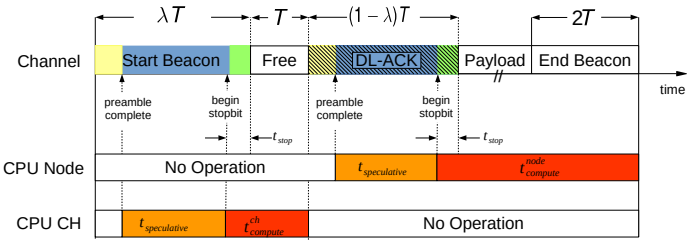

Fig. 8. Channel and computation utilization from Node and CH. The task of computing updates for the clocks after a successful beacon receivement has to be complete after time $t^{ch}_{compute}$, $t^{node}_{compute}$, respectively.

TABLE II
Microcontroller Implementation Parameters

Parameters	Value
Frequency	2.4GHz
Bandwidth	2Mhz
Datarate	250kbit/s
Microcontroler clock	7.3728MHz
PCO-Period T	10ms
Coarse Clock Period LT	1.2s
$(\alpha, D, \delta, \beta, \lambda)$	(0.125, 15, 7, 0.7, 0.5)

In Section IV we stated that PulseSS requires acknowledgments from multiple CH to be sent at unison so that the receiver will view them as a multi-path channel transmission. In our implementation we only have access to the MAC and not the physical layer. As a consequence, the accuracy achieved is not sufficient for multiple CH's acknowledgments to be received in times that are close enough to be considered as multi-path (are not within the cyclic prefix of the OFDM transmission). Therefore we extended the PCO period T so that each CH uses a deterministic lag with respect to their own clock that prevents the collisions with neighboring CH.

In the first experiment we have 2 clusters with one shared node and 3 and 2 local nodes respectively, as seen in the top right of Fig. 9. To control the topology we use tin-foil to create Faraday-Cages that would result in the desired connectivity.

In Fig. 9, we plot the PM between each node in the network and CH 1, chosen as the reference. We can see that, apart for the shared node PM (red line) the PM of the nodes in cluster C1 and C2 are very close, which means that the nodes tend

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

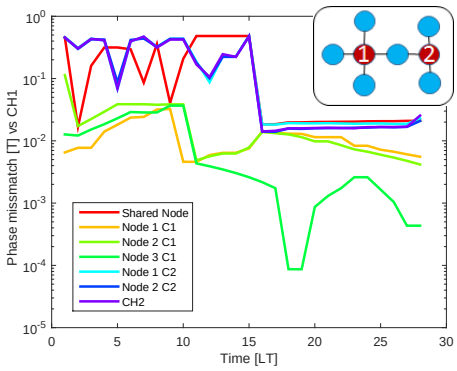

Fig. 9. Synchronization result and topology.

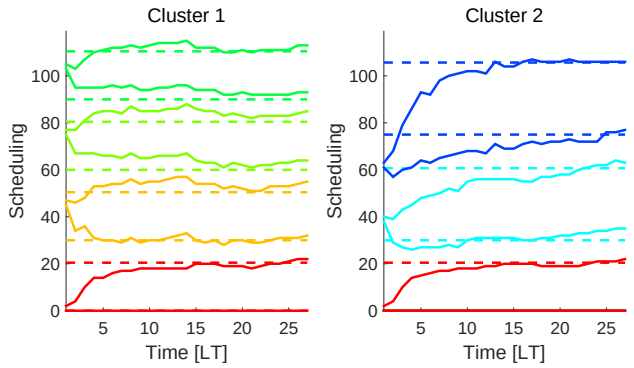

Fig. 10. TDMA Scheduling result of topology in Fig 9. Note that the shared node, in red, is similar but not identical in both clusters. This is due to independent recording of the beacon arrival times in the 2 CH, with respect to their own clock.

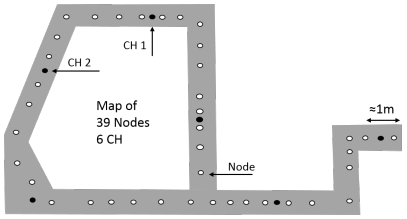

Fig. 11. Deployment Map of 39 Nodes and 6 CHs.

to lock their PCO clocks in their respective clusters, while the shared node, in red, is initially oscillating between the two clusters, causing oscillations. After 16 cycles of the coarse clock (LT) the system converges to a common timing, confirming that, via shared nodes, the clusters are lead to coalesce to the same network timing. With the given parameters in Table II, we computed the average error after convergence in the experiment to be 80μs, which is significantly larger than the simulation results. Recall that we do not have access to the physical layer and therefore cannot estimate beacon arrival times precisely.

We also noticed that this system is converging a lot faster than in the simulations, which is reasonable since our α (Table II) is larger by an order of magnitude, speeding up convergence. Another reason to chose α larger is the limited precision of 16-Bit fix point numbers compared to 64-Bit floating point numbers used in the simulation[7]. Similarly, we chose the parameter β for the scheduling updates larger than in the simulations, to speed up scheduling convergence. From the simulation we found that β to be between 0.2 and 0.7 yields good results.

The system's TDMA scheduling result can be seen in Fig. 10, showing the theoretical solution in dashed lines and the measurement result in solid lines. We can see that the system is converging to the theoretical solution and oscillates around it due to the dithering. We can see each node obtaining its proportional fair schedule in this experiment. Specifically the shared node in red, is correctly associated with the denser cluster 1, as we would expect from our association rule.

In our second experiment we tested the same exact implementation of PulseSS, but on a much larger set of 39 nodes and 6 CHs. We also have the same set of parameters than in the previous experiment, except for T set to 50 ms. The network deployment shown in Fig. 11 was inside a hallway. The conflict graph was determined by the nodes position and interference. As before CH's were preassigned to nodes. We can see that scheduling aligns and shared nodes are distributed among the available space as seen exemplary for 2 of the 6 clusters in Fig. 12 and 13. We notice that group A in the top cluster is at the edge of reception and loses connection after some time. Our interpretation is that as transmission begins and the battery powered node begins to consume more power, the battery voltage drops, resulting in reduced transmission range. The network simply adjusts to occupy the slots that are available after these nodes fail. We notice 5 shared nodes (group B) that are present in both clusters. We record the schedule attained locally on each CH with respect to the CH's own PCO clock, thus shared nodes are only similar and not identical for both clusters and the schedules appear as drifting, because of frequency drift and rounding errors of the CH's clocks. We can further see that the conflicts of the nodes in cluster 1, which is the densest cluster, ultimately determine the schedule. This experiment is described in more detail in [1]. We compared the outage rate of ALOHA, CSMA and PulseSS and found that PulseSS, which with its signaling ensures the channel is only in use by one node at a time, is the most successful in avoiding conflicts. In fact, in comparing the number of failed packet transmissions we find that the failure rates of ALOHA, CSMA and PulseSS are 39.5%, 23.5% and 8% respectively for the same exact deployment and the same exact traffic pattern. This is not surprising, since CSMA works best for bursty traffic, while PulseSS is most suited for the type of continuous traffic that we generated. In addition, we found for this experiment that the average data throughput per node of PulseSS was 14kbit/s with an average protocol overhead of 2.1% resulting in an average channel usage of 69%.

Through this second experiment we were able to confirm that the protocol works in this large scale networks in the way we predicted in theory. As far as the synchronization performance are concerned, in this second experiment we obtained an average synchronization accuracy of 410μs in Fig. 14. As expected, the accuracy degrades compared to our previous experiment, as a consequence of errors adding up

[7]The change of phase with a small α would always be rounded to 0.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2016.2576923, IEEE Internet of Things Journal

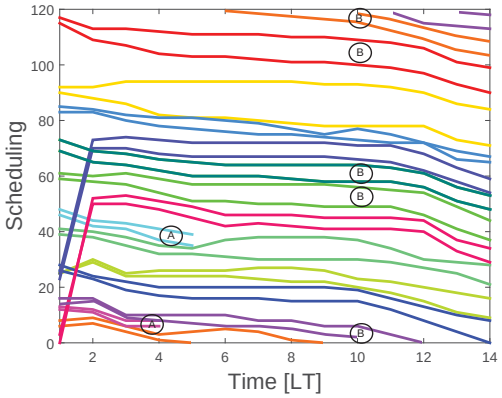

Fig. 12. TDMA Scheduling result of cluster 1.

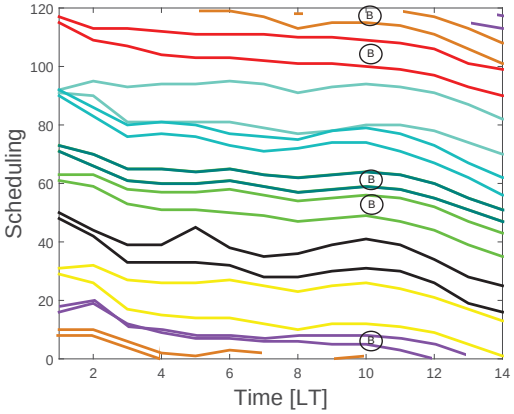

Fig. 13. TDMA Scheduling result of cluster 2.

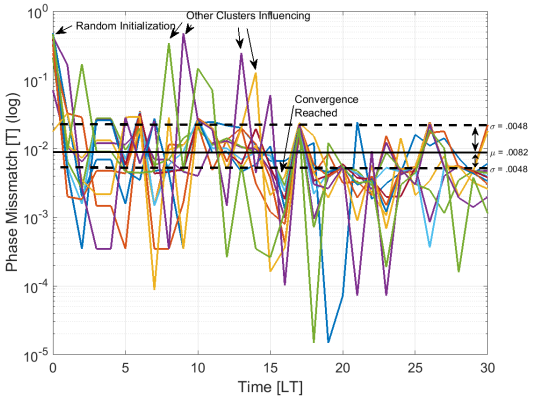

Fig. 14. Synchronization result of both shown clusters. Mean and standard deviation are calculated from the point of convergence.

over more hops in this larger network scenario.

VII. Conclusions

In this paper, we proposed PulseSS, a protocol that provides network synchronization and proportionally fair scheduling in wireless mesh-networks with a clustered structure. The protocol was loosely inspired by the PCO model from mathematical biology. PulseSS' main appeal is that of providing scheduling and synchronization functionalities by exploiting simple physical layer signaling and local network updates. The complexity of the updates remains unchanged as the size of the network scales up, offering a competitive alternative for wireless sensor networks to main-stream protocols, like WirelessHart, especially in applications that are delay sensitive and need a resilient clock distribution mechanism, e.g. Intelligent Infrastructures, Internet of Things, Control Area Networks and Cyber-Physical Systems.

References

[1] R. Gentz, A. Scaglione, Y.-W. Hong, and L. Ferrari, "Pulsess: A microcontroller implementation of pulse-coupled scheduling and synchronization protocol for cluster-basedwireless sensor networks," 2015, iEEE World Forum on Internet of Things.

[2] S. Ramanathan, "A unified framework and algorithm for (t/f/c)dma channel assignment in wireless networks," in *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, Apr 1997, pp. 900–907 vol.2.

[3] X. L. Huang and B. Bensaou, "On max-min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2001, pp. 221–231.

[4] C. D. Young, "Usap: a unifying dynamic distributed multichannel tdma slot assignment protocol," in *Military Communications Conference, 1996. MILCOM'96, Conference Proceedings, IEEE*, vol. 1. IEEE, 1996, pp. 235–239.

[5] L. C. Pond and V. O. Li, "A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks," in *Military Communications Conference, 1989. MILCOM'89. Conference Record. Bridging the Gap. Interoperability, Survivability, Security., 1989 IEEE*. IEEE, 1989, pp. 70–74.

[6] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. CUller, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless networks," *Proceeding of SenSys*, 2007.

[7] 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.

[8] 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.

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

[10] T. Lennvall, S. Svensson, and F. Hekland, "A comparison of wirelesshart and zigbee for industrial applications," *IEEE Workshop on Factory Communication Systems*, 2008.

[11] I. An, "Standard wireless systems for industrial automation: Process control and related applications, isa std," *ISA-100.11 a-2009*, 2009.

[12] 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.

[13] 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.

[14] A. Motskin, T. Roughgarden, P. Skraba, and L. Guibas, "Lightweight coloring and desynchronization for networks," in *INFOCOM*. IEEE, 2009, pp. 2383–2391.

[15] H. Kang and J. L. Wong, "A localized multi-hop desynchronization algorithm for wireless sensor networks," in *INFOCOM*. IEEE, 2009, pp. 2906–2910.

[16] Y.-W. Hong and A. Scaglione, "Time synchronization and reach-back communications with pulse-coupled oscillators for uwb wireless ad hoc networks," in *IEEE Conference on Ultra Wideband Systems and Technologies*, 2003, pp. 190–194.

[17] R. Pagliari and A. Scaglione, "Scalable network synchronization with pulse-coupled oscillators," *IEEE Trans. Mobile Computing*, vol. 10, no. 3, pp. 392–405, 2011.

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

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

[20] 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*. ACM, 2005, pp. 142–153.

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

[22] L. Ferrari, A. Scaglione, and Y.-W. Hong, "Convergence results on pulse coupled oscillators protocols in locally connected networks," 2015, submitted to IEEE Transactions on Networking.

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

[24] Y. Wang and F. J. Doyle, "Optimal phase response functions for fast pulse-coupled synchronization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5583–5588, 2012.

[25] Y. Wang, F. Nunez, and F. J. Doyle, "Energy-efficient pulse-coupled synchronization strategy design for wireless sensor networks through reduced idle listening," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5293–5306, 2012.

[26] ——, "Statistical analysis of the pulse-coupled synchronization strategy for wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5193–5204, Nov 2013.

[27] A. Tyrrell, G. Auer, and C. Bettstetter, "On the accuracy of firefly synchronization with delays," in *Applied Sciences on Biomedical and Communication Technologies, 2008. ISABEL '08. First International Symposium on*, Oct 2008, pp. 1–5.

[28] L. Ferrari, R. Gentz, A. Scaglione, and M. Parvania, "The pulse coupled phasor measurement units," *IEEE Smartgridcomm*, 2014.

[29] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. c1–269, Jul 2008.

[30] D. Dardari and M. Z. Win, "Ziv-zakai bound on time-of-arrival estimation with statistical channel knowledge at the receiver," in *IEEE International Conference on Ultra-Wideband, 2009.* IEEE, 2009, pp. 624–629.

[31] 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.

[32] 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.

[33] S. Ashkiani and A. Scaglione, "Discrete dithered desynchronization," *arXiv preprint arXiv:1210.2122*, 2012.

[34] C. R. Anderson and T. S. Rappaport, "In-building wideband partition loss measurements at 2.5 and 60 ghz," *Wireless Communications, IEEE Transactions on*, vol. 3, no. 3, pp. 922–928, 2004.