PulseSS: A Microcontroller Implementation of Pulse-Coupled Scheduling and Synchronization Protocol for Cluster-Based Wireless Sensor Networks

Reinhard Gentz, Anna Scaglione, Lorenzo Ferrari
Electrical Engineering Department
Arizona State University
Tempe, USA
{*rgentz,anna.scaglione,lferrari*}*@asu.edu*

Y.-W. Peter Hong
Institute of Communications Engineering
National Tsing Hua University
Hsinchu, Taiwan
ywhong@ee.nthu.edu.tw

***Abstract*—We propose a new protocol, named Pulse coupled Synchronization and Scheduling (PulseSS), inspired by the dynamics of pulse coupled oscillators (PCO). PulseSS coordination signaling mechanism provides both decentralized network synchronization and time division multiple access scheduling for clustered networks. Contrary to previous PCO based protocols proposed for scheduling, PulseSS can resolve conflicts in locally connected networks, enabling mesh infrastructures to work synchronously using this protocol. At the same time the protocol retains the adaptivity and light-weight nature of PCO protocols both in terms of signaling as well as computations. In the paper we demonstrate the efficiency of PulseSS for TinyOS systems, with an outage rate of only 8.0%, compared to 23.5% for traditional CSMA-CA, while also providing time synchronization across the network.**

***Keywords*-Pulse coupled oscillators, distributed time scheduling, clustered networks.**

I. Introduction

Wireless sensor networks (WSN) deployments most often rely on WiFi or Zigbee radios, which are inherently asynchronous and resolve access conflicts through Carrier Sensing Multiple Access (CSMA). Typically the need for network synchronization is addressed through out-of-band control channels or message exchanges, with timing coming prevalently from a Global Positioning System (GPS) receiver, or through a clock distribution network protocol, such as the Precision Time Protocol (PTP). While these solutions can supply time information for the synchronous sampling, they do not solve the sensor scheduling problem which can be important in applications where data are produced and need to be remotely delivered at a regular pace.

In this paper we propose a lightweight and easy to deploy protocol for clustered ad-hoc networks, combining decentralized synchronization and medium access control, named the Pulse coupled Synchronization and Scheduling (PulseSS) protocol. PulseSS works in an *ad-hoc* mesh network scenario, where nodes are grouped into clusters and contend for the same spectrum resources adaptively. Each cluster has a special node acting as cluster head (CH), with similar properties 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, so that hidden terminals can learn about conflicts.

What truly distinguishes the protocol from standard carrier sensing solutions or other network synchronization protocols, is the way the events are processed and how permission for medium access is granted. In fact, the protocol attains synchronization and scheduling information through network state dynamics that are loosely inspired by the Pulse Couple Oscillators (PCO) [1] model, and therefore fall in the category of bio-inspired designs.

A popular centralized protocol to similarly attain synchronization and scheduling with widespread acceptance for WSN is WirelessHART [2]. Scheduling in WirelessHART is centrally managed by a dedicated node called 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. Second, 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.

A. Related research

As there are many scheduling and synchronization algorithm published, we present here only those relevant to our approach. Prior related examples of PCO-based scheduling algorithms are the DESYNC protocol in [3], and our previous work on Proportional Fair Scheduling algorithm [1], and follow up work [4], [5] that relaxes the all-to-all connectivity assumption, required by the former two.

There is also a vast literature on PCO synchronization, see e.g. [6]–[9]. However, PCO synchronization does not work well if it is merged with CSMA/CSCA protocols [10]. Efficient implementations of the PCO protocol disable CSMA [7] or use a separate radio band all-together [11]. 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. Compared to [3] and [1] the scheduling protocol we propose resolves conflicts among neighboring cells and does

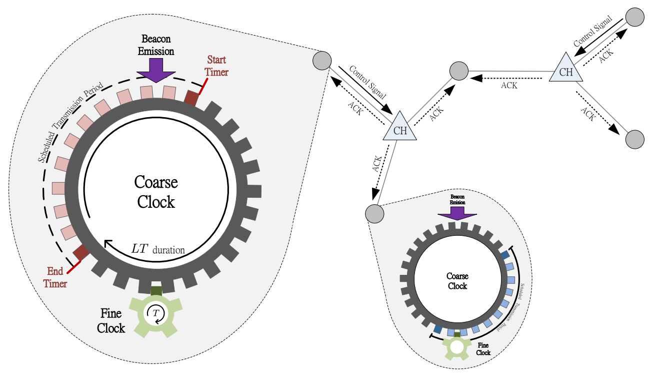

Figure 1. The coarse and fine clocks maintained by the nodes.

not require an all-to-all connectivity. In fact, the PulseSS protocol can be viewed as the realization and integration of the theory of PCO synchronization and desynchronization previously studied in e.g. [1], [3]–[9]. The benefits that we will showcase are: 1) collision avoidance; 2) all-in-one signaling; 3) potential for greater synchronization accuracy.

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 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 with the CHs. For each $c \in \mathcal{C}$, we define $\mathcal{N}_c \subset \mathcal{V}$ as the set of non-CH nodes that lie within the transmission range of CH c; and, for each $v \in \mathcal{V}$, 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 there is a predefined association between nodes and cluster-heads that is compatible with their communication range and that every c has an associated $\mathcal{V}_c \subset \mathcal{N}_c$ forming a partition of $\mathcal{V} = \cup_{c \in \mathcal{C}} \mathcal{V}_c$. The nodes that are neighbors of multiple CHs are referred to as the *shared* (or *gateway*) *nodes*. The management of these nodes is crucial to ensure that neighboring clusters can self-organize to attain conflict free schedules.

In PulseSS, each node maintains the phase of two local continuous clocks, which we call *state*: the phase of a *fine clock* with period T and the phase of 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 while the coarse clock period represents a frame with L time slots. The PulseSS signaling is used to locally update the phases of both clocks. These updates synchronize the phases of the fine clocks at all nodes (i.e., synchronizes all nodes at the slot level) and sets the phases of the coarse clocks apart so as to schedule for each node a portion of the L time slots of the coarse clock, enabling proportional fairness and spatial reuse. These goals are achieved by having each node fire a pulse (i.e. transmit a control signal or preamble which we call the *beacon*) to neighboring CHs every time a cycle of its local coarse clock expires, and by having other nodes adjust their clocks or schedules as the CHs' acknowledgment of the pulse is heard. The notion of being coupled through an acknowledgment, widely used in random access protocols, is new in PCO based protocols, and it is the key ingredient to attain coupling and collision avoidance at once, while retaining the simplicity and scalability of PCO signaling.

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 cloak, i.e., a *start timer* and an *end timer*, as depicted in Fig. 1. The timers delimit the time in which the node can transmit information to its CH. The expiration of the timers marks the first and last time slots that node v is scheduled to transmit in. The state of the start and the end timers can be described by the phase variables

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

$$\begin{aligned}\Psi_v^{(e)}(t) &\triangleq \frac{t}{T} + \phi_v + \psi_v^{(e)} \pmod L \\ &= I_v^{(e)}(t) + \Phi_v(t), \end{aligned} \tag{3}$$

where $\psi_v^{(s)}$ and $\psi_v^{(e)}$ are integer offsets of the timers and $I_v^{(s)}(t) \triangleq \lfloor \Psi_v^{(s)}(t) \rfloor$ and $I_v^{(e)}(t) \triangleq \lfloor \Psi_v^{(e)}(t) \rfloor$ are the indices of the start and end time slots. The timers expire when their respective phase variables reach the value L and reset to 0 afterwards. Note that $[\Psi_v^{(s)}(t) - \Psi_v^{(e)}(t) \pmod L] = [I_v^{(s)}(t) - I_v^{(e)}(t) \pmod L]$ indicates the number of time slots scheduled to node v. Two control signals are sent between node v and its CH at the start and end of a scheduled transmission period and payload data is sent in between. The two signals are called the *start* and *end beacons*. The corresponding acknowledgments are called *start* and *end acknowledgments*. The scheduling is achieved by updating the discrete portions of the start and end timers (i.e., $I_v^{(s)}(t)$ and $I_v^{(e)}(t)$, by overhearing the acknowledgments sent by the CHs in range, as explained in Section IV) whereas the synchronization is achieved by updating the phase $\Phi_v(t)$ (discussed in Section III). Nodes and CHs transmit identical beacons; they essentially constitute a common PHY layer preamble and post-amble encapsulating their respective message, which PulseSS utilizes to perform its MAC function, reducing significantly the MAC layer overhead. Furthermore, these messages do not have to be decoded, they simply have to be detected, performing a multi-hypothesis test with four hypotheses.

A tedious but important fact to remark is that in PulseSS the clock dynamics (and schedule) of a node $v \in \mathcal{V}_c$ are not only a function of the feedback received by its CH c, but also a function of acknowledgments of neighboring CHs, that can exert pressure coming from their own cluster. While the CH of node v needs to confirm if *the channel is clear for transmission*, so that collisions with hidden nodes are avoided, schedule updates of node v follow the firing of any CH in its range that happened right before and right after node v's start and end clock expire, respectively. Hence, node $v \in \mathcal{V}_c$ may update its clocks reacting to another CH c'. In this case, we say that node v updates with CH c'.

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:** (CA) 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:** (PFS) The transmission schedules of all nodes associated with the same cluster are proportional to their demands.

Moreover, in each time slot, we divide the duration T into two parts: an uplink transmission period with duration $T_u = \lambda T$, $\lambda \in (0,1)$, and a downlink transmission period with duration $T_d = (1-\lambda)T$. 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 in range of each other but are transmitting and receiving simultaneously with different partners. Hence, in PulseSS (like in any protocol with acknowledgements used for collision avoidance) the hidden terminal problem is naturally resolved through the acknowledgment beacons by the CH's.

III. PulseSS Synchronization Updates

Now we describe the clock update procedures required to achieve synchronization among all fine clocks in the network. Note that the synchronization relies only on the beacons and acknowledgements utilized for scheduling and, thus, comes at *no additional signaling overhead.*

A. Effect of Delay on Conventional PCO Synchronization

Suppose that a beacon was emitted by node u at time t, and that every other node $v \neq u$ who is not firing at the same time, updates its local phase variable so that the value at time t^+ becomes

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

By doing so, the local phase variable is advanced by an amount proportional to its current value. Under the ideal assumptions that the pulse is received with no delay and all-to-all connected network, has been shown in [12], and will eventually converge to synchrony with probability 1 as time goes to infinity. That is, there exists t_0 such that $\Phi_v(t) = \Phi_u(t)$, for all nodes u, v and for all $t > t_0$. In the literature, PCO convergence in arbitrary locally connected networks was proven by considering asymptotically small coupling (i.e. small α) [8], [9].

Note also that to allow for propagation delay, several works 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. 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. Therefore, the duration Δ_{ref} of the refractory period must be chosen to be larger than the propagation delay and the likely range in which the estimation error will be. Let r be the time of reception of a pulsing event that occurred at time t modified update rule is given by

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

While, the size of the refractory period that is necessary for the protocol convergence must ensure that $r - t < \Delta_{\text{ref}}$ for all neighborhoods in the network, in locally connected networks the accuracy of PCO synchronization can be actually worse than the worst delay error. In fact, in [13] it was shown that the error accumulates in general over multiple hops and, thus, can be in general bigger than the Δ_{ref} required to observe convergence. PulseSS can mitigate this effect by estimating the signal traveling times from the bounced back acknowledgments [14] and compensate for them in the update. However for our implementation we are not using this feature yet, as the digital controller in our implementation runs with a clock period much lower than the round-trip delay.

IV. PulseSS Scheduling Updates

The scheduling of PulseSS for CA and PFS is achieved through the update of the discrete portions of the nodes' start and end timers (i.e., $I_v^{(s)}(t)$ and $I_v^{(e)}(t)$) in each cycle of the coarse clock. The update and messaging mechanisms can be viewed as the realization of the theory of PCO desynchronization studied in [1]. 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 $u, v \in \mathcal{N}_c$,

$$\Psi_v^{(s)}(t) - \Psi_v^{(e)}(t) \leq \Psi_v^{(s)}(t) - \Psi_u^{(s)}(t) \pmod{L}. \qquad (6)$$

This can be achieved by letting the initial difference of the start and end timers at each node be sufficiently small, accompanied by admission control at the CH as to be discussed later. The expiration of the start and end timers marks the start and end of node v's transmission period in each cycle. Once the start (or the end) timer expires, a start (or an end) beacon is emitted by node v in the UL period of the time slot. These beacons signals will then be acknowledged by all CHs in range of node v inform all other nodes of the beacon emission by node v. Recall that $\mathcal{C}_v$ is the set of CHs in range of node v, and let $\mathrm{pre}(v) \in \mathcal{C}_v$ and $\mathrm{suc}(v) \in \mathcal{C}_v$ 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 timer of node $\mathrm{pre}(v)$ and the start timer of node $\mathrm{suc}(v)$.

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. 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 acknowledgements to these beacon signals, but the accuracy may be affected by synchronization errors and propagation delays. The time estimates at node v are denoted by $\hat{t}_{\mathrm{pre}(v),v}^{(e)}$ and $\hat{t}_{\mathrm{suc}(v),v}^{(s)}$.

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

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

where D_v is the demand of node v, δ is the portion of time slots reserved as guard period in between transmissions, and $t^+ \triangleq \lim_{\epsilon\to 0} t+\epsilon$ indicates the time immediately following t. 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

$$\hat{I}_{v,\mathrm{target}}^{(s)} = \frac{D_v+\delta}{D_v+2\delta} \frac{\hat{t}_{\mathrm{suc}(v),v}^{(s)+} - \hat{t}_{\mathrm{pre}(v),v}^{(e)}}{T} \tag{7}$$

$$\hat{I}_{v,\mathrm{target}}^{(e)} = \frac{\delta}{D_v+2\delta} \frac{\hat{t}_{\mathrm{suc}(v),v}^{(s)+} - \hat{t}_{\mathrm{pre}(v),v}^{(e)}}{T}. \tag{8}$$

Since the target values are not precise[1], 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 schedules. This is achieved by further modifying their target values as

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

$$\tilde{I}_{v,\mathrm{target}}^{(e)} = \max\left\{I_{v,\mathrm{target}}^{(e)}, \frac{I_v^{(e)}(\hat{t}_{\mathrm{suc}(v),v}^{(s)+})}{2}\right\}.$$

Finally, the local timers at node v are updated as

$$I_v^{(s)}(\hat{t}_{\mathrm{suc}(v),v}^{(s)+}) = \mathcal{Q}\left[(1-\beta) I_v^{(s)}(\hat{t}_{v,\mathrm{suc}(v)}^{(s)}) + \beta \tilde{I}_{v,\mathrm{target}}^{(s)}\right] \tag{9}$$
$$I_v^{(e)}(\hat{t}_{\mathrm{suc}(v),v}^{(s)+}) = \mathcal{Q}\left[(1-\beta) I_v^{(e)}(\hat{t}_{\mathrm{suc}(v),v}^{(s)}) + \beta \tilde{I}_{v,\mathrm{target}}^{(e)}\right] \tag{10}$$

where $\beta \in (0,1)$ and $\mathcal{Q}(\cdot)$ is a dithered quantization function [15] that maps the phase to the integer set $\{0,1,\ldots,L\}$ defined as $\mathcal{Q}(x) = \mathrm{round}(x+v)$, where $v \sim \mathcal{U}(-1/2,1/2)$. As shown in [16], the dithering is added because it ensures the convergence of an average quantized consensus policy. Notice that, 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 [17]. *Global fairness* can be attained under certain necessary and sufficient assumptions on the network conflicts that lead the protocol to converge to unique conflict free-schedule (irrespective of the initial conditions). Under a random deployment and initialization like the ones we tested these conditions may not be met, but depending on the random initial conditions, the nodes will reach a variety of conflict free schedules, that are proportionally fair locally, but may leave empty slots.

V. Microcontroller Implementation

To evaluate the PulseSS algorithm, we first did extensive simulations to verify our ideas. Followed up with an implementation of the protocol in TinyOS operating system, in order to test real world applicability. Our implementation runs on the Crossbow MicaZ platform with an 7.3728Mhz Atmel AtMega128L and a CC2420 Radio. The microcontroller does not have access to the physical layer but MAC layer only, because of limitations of the used CC2420 Radio chip. This limits us to send only OFDM packets with fixed cyclic prefix. When scheduling a shared node the task of making each CH respond identically and simultaneous is critical, however our accuracy in timing is lower than the for this platform unchangeable cyclic prefix, thus transmission from multiple sources (with the same content), that should look like one multipath transmissions to the receiver, fail as signals are colliding and are lost. To mitigate this effect we lengthen the PCO Period T such that each CH can repeat its ACK signal in its own dedicated time slot.

[1]They are based on partially outdated information about the successor nodes states revealed when it last fired.

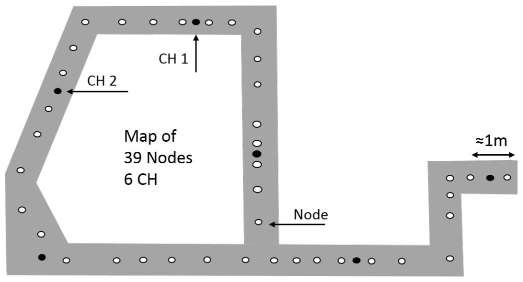

Figure 2. Deployment map in a hallway

Parameters	Value
Frequency	2.4GHz
Bandwidth	2Mhz
Datarate	250kbit/s
Microcontroler clock	7.3728MHz
PCO-Period T	50ms
Coarse Clock Period LT	6s
(α,D, δ,β,λ)	(0.125,15,7,0.7,0.5)

Table I
PARAMETERS USED IN THE PULSESS IMPLEMENTATIONS

We implemented our algorithm on a test-network of 39 Nodes and 6 CH. We compare the throughput of our protocol versus the default protocol in TinyOS *Carrier Sensing Collision Avoidance* (CSCA) and pure ALOHA random access. For this comparison we let each node transmit sequential data to its CH and check the rate of success. In PulseSS a node transmits data, if and only if it has transmitted its start beacon, but not yet its end beacon. Using the parameters from Table I we can find the channel usage to be 69%. To emulate a similar channel usage using random access each node waits a random time between transmissions, using the build-in uniform pseudo random number generator, such that it matches the channel usage of PulseSS. For our experiment we transmit 250 packets per node once converged. We notice, in Table II, that the failure rate increases with the number of nodes communicating, as there is a higher chance of two nodes overlapping, i.e. are colliding. In the 6 CH case random access approaches 39.5% packet loss, CSCA 23.5% and PulseSS is only 8.0%, as the protocol ensures that only one node transmits at a time per cluster. We can further see that even if there is only one node in the cluster, and no interference/outage should occur, there is an outage rate of $\approx 1.6\%$ for all MACs. We suspect that this is caused noise, likely caused by nearby 2.4GHz devices transmitting at the same frequency when we carry out the experiments, while for large networks the network self-interference, caused by our devices, is dominant. The achieved results show that PulseSS is especially suited for maximizing throughput per area unit, as self-interference in the network is minimized.

PulseSS converges after some iterations, until all nodes agree to a PCO, fine clock time. The accuracy archived in our test network averages to 410μs after converges after 16 cycles as seen in Fig.3. While these are good results for a network of this size, we have to pay attention to the following: recall that we can only send packets but have no control of the physical layer firmware. Unfortunately, this

One CH + Nodes	ALOHA-Fail	CSMA-Fail	PulseSS-Fail
1	1.6%	1.6%	1.6%
2	5.4%	2.2%	1.6%
3	11.1%	2.3%	1.4%
4	12.8%	2.9%	2.0%
Six CH + Nodes			
39	39.5%	23.5%	8.0%

Table II
COMPARISON OF TRANSMISSION FAILURE RATES

error is not within the cyclic prefix (CP) of the OFDM PHY signal. With greater accuracy, we expect that the concurrent CH acknowledgments can be treated at the receiver as a signal affected by multi-path. In our case, however, they fall outside the OFDM CP and therefore the nodes experience collisions among acknowledgments. For this experiment we assign each CH to its own time-slot by giving each a fixed delay to send their ACK beacon, such that collisions do not occur. We further notice that computation limitations in the microcontoller, i.e. the controller might be busy when a packet arrives. As it is the controllers duty to record packet arrival time, the workload of the controller affects the time accuracy, and we can only compensate for the average extra delay we experience. In addition we face storage limitations. To deal with them we had to record synchronization and scheduling results in two independent sessions.

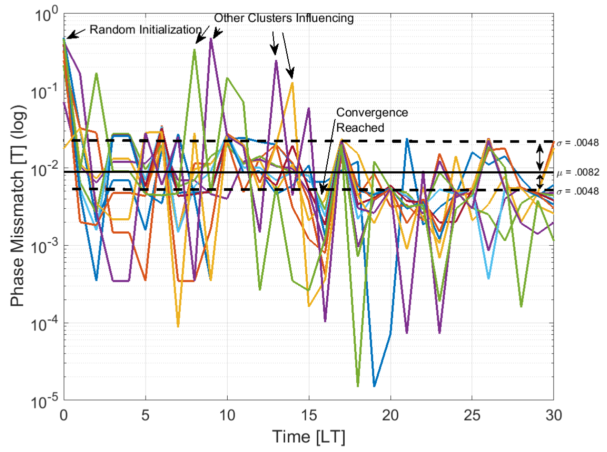

Figure 3. Synchonization result of both shown clusters. Mean and standard deviation are calculated from the point of convergence.

For scheduling we record each nodes start and end counter, shown color-coded for each node over time in Fig. 4. We can see that scheduling aligns and shared nodes are distributed among the available space as seen exemplary for 2 of the 6 clusters. The jumps in the figure are due to the fact that the schedule is modulo L, thus 0 and L-1 are adjacent. It is also interesting to note that group A in the top cluster are at the edge of reception and lose reception after some time. Our interpretation is that as transmission begins and the battery powered node begins to consume power, the battery voltage drops, resulting in reduced transmission range. The network simply adjusts to occupy the frame that is available after these nodes fail. In both clusters we can identify 5 shared nodes, group B, that are present in both clusters. As the results are recorded by each CH with respect to its own PCO clock, shared nodes are only similar and not identical for both clusters and the schedules appear as drifting, because of frequency drift and rounding of CH's clocks. We can further see in the top cluster, that there are more nodes in the cluster thus squeezing the shared nodes schedule compared to the nodes that are only in the bottom cluster. We further notice that we do not archive global fairness, due to random initialization, as a certain initialization is required and are discussed in [14]. Nevertheless node C and node D make the best out of the situation and extend themselves to take advantage of the interference free time available to reach their respective CH. Instead, node E right below node D is again limited by another node from a third cluster and thus both node D& E do not share the space equally. To attain greater efficiency the CH noticing inefficiencies could avoid acknowledging other nodes forcing them to reconnect in another position in the frame. The node then has a new random chance of picking the slot which would ultimately lead to global fairness. However, this would complicate the protocol and, as we noticed, there are already significant gains to be ripped from PulseSS even in this partial implementation.

We presented novel decentralized algorithm that provides both synchronization and scheduling with a simple signaling mechanism. For future work we intend to switch from a microcontroller to an FPGA platform, for direct access to the physical layer, allowing shaping of beacons and higher clock rates, resulting in higher accuracy and and a reduced of T (i.e. shorter latency).

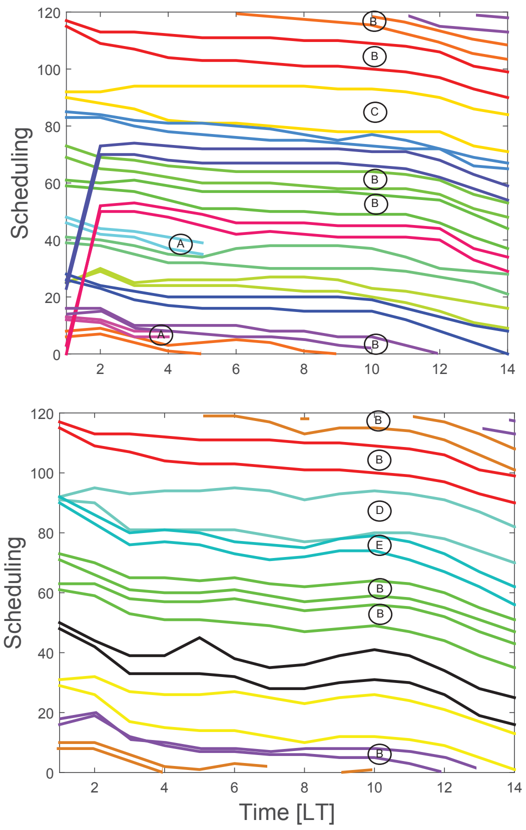

Figure 4. Scheduling result of 2 clusters. Top cluster 1; Bottom cluster 2

References

[1] 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.

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

[3] 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.

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

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

[6] 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.

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

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

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

[10] 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.

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

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

[13] 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.

[14] R. Gentz, P. Hong, L. Ferrari, and A. Scaglione, "Pulsess: A pulse-coupled scheduling and synchronization protocol for cluster-based wireless sensor networks," in preparation.

[15] 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.

[16] 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.

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